MASTER EN SCIENCES INFORMATIQUES

Traitement d'images couleur
localisation d'objets en temps réel

André, Arnaud; Frippiat, Sébastien

Award date:
2005

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'informatique
Année académique 2004-2005

Traitement d'images couleur : Localisation d'objets en temps réel

Arnaud André Sébastien Frippiat

Mémoire présenté en vue de l'obtention du grade de Maître en informatique

Résumé

Le traitement d'images couleur est un domaine très investigué par les scientifiques ces dernières années. Un sous-ensemble de ces recherches concerne l'asservissement visuel. Le groupe TROP de l'université de Haute-Alsace, en France, s'intéresse activement à ce sujet et a développé un algorithme, le « Colour Histogram Similarity » (CHS), qui permet de localiser une cible dans une image couleur. Leurs contraintes sont un faible temps de calcul et un minimum d'hypothèses sur les images analysées.

Ce mémoire va se concentrer sur l'étude du comportement du CHS et de ses faiblesses qui sont notamment l'illumination de la scène et la taille de la cible. Il comprendra l'analyse détaillée de ces problèmes et des solutions seront proposées. Le choix de l'espace couleur ainsi que la quantification de la qualité des résultats du CHS seront également abordés.

Mots-clés : asservissement visuel, traitement d'images, localisation couleur, espaces couleur, histogramme, « Colour Histogram Similarity », CHS, temps réel, robustesse à l'illumination, estimation de taille, validité des résultats.

Abstract

Colour image processing is an area which has been widely investigated by scientists for the last few years. A subset of these researches deals with visual servoing. The group TROP from the university of Haute-Alsace, in France, has been actively working on this subject and has developed an algorithm, the « Colour Histogram Similarity » (CHS), which allows locating a target in a colour image. The constraints are a low calculation time and a miminum of assumptions on the analysed images.

This master's thesis will focus on the study of the behaviour and weaknesses of the CHS which are among others the scene illumination and the target size. It will include a detailed analysis of these problems and solutions which will be proposed. The choice of the colour space and the quantification of the quality of the results of the CHS will also be presented.

Keywords : visual servoing, image processing, colour localisation, colour spaces, histogram, Colour Histogram Similarity, CHS, real time, illumination robustness, size estimation, results validity.

Nous tenons à remercier les professeurs J.-L. Buessler et J.-P. Urban pour l'accueil qu'ils nous ont réservé au sein du laboratoire TROP pendant les cinq mois de notre stage. Ils ont consacré énormément de temps à suivre notre travail et nous ont fourni de nombreux conseils. Nous désirons leur faire part de notre gratitude pour avoir bien voulu relire l'entièreté de ce mémoire et nous avoir proposé des pistes pour l'améliorer.

Nous remercions également notre promoteur, J.-P. Leclerq, pour la lecture critique et constructive de notre mémoire, de même que pour ses encouragements durant sa rédaction.

Nous remercions nos familles ainsi que Sévïn et Delphine pour le soutien qu'elles nous ont apporté durant le stage et pour les relectures de ce mémoire.

Finalement, nous remercions Djaffar, Gilles et les autres membres de l'équipe du TROP pour leur accueil et l'ambiance décontractée du laboratoire. Un merci particulier à Djaffar pour le café qui agrémentait les pauses.

Table des matières

Introduction

Dans le cadre de ce mémoire, nous avons été accueillis au sein du laboratoire TROP de l'université de Haute-Alsace, à Mulhouse. L'équipe du TROP étudie le problème de l'asservissement visuel de processus robotiques. L'asservissement visuel consiste à contrôler un robot en se basant sur des informations visuelles. Pour cela, le TROP envisage des approches innovatrices telles que l'utilisation d'algorithmes neuro-mimétiques et des techniques de traitement d'images couleur.

Notre sujet de mémoire, la localisation d'objets en temps réel, s'inscrit dans la prolongation des travaux réalisés par le TROP. Un travail de DEA a notamment été réalisé sur ce sujet et a été présenté peu de temps après notre arrivée à Mulhouse. Durant notre stage, nous nous sommes affranchis des contraintes matérielles car, lors de l'élaboration d'un nouvel algorithme, il faut soigneusement vérifier tous les cas de figure avant de penser à une intégration robotique. Une méthode de localisation couleur, le « Colour Histogram Similarity », est activement utilisée par le laboratoire et nous avons participé à son amélioration.

L'approche envisagée par le TROP est originale dans le sens où des contraintes très strictes sont employées. Ainsi, les techniques utilisées doivent permettre une utilisation en temps réel, c'est-à-dire qu'il faut pouvoir traiter au moins 25 images par seconde. De plus, aucune hypothèse n'est émise sur les conditions d'acquisition des images. En d'autres termes, les techniques ne doivent pas être sensibles à l'illumination, la taille de la cible, le décor, ...

Nous allons étudier, tout au long de ce mémoire, la problématique d'une localisation basée uniquement sur des informations de couleur. Pour cela, nous commencerons par détailler notre environnement et notre démarche de travail au sein du laboratoire TROP. Cette démarche consiste à rechercher une cible, définie au préalable, dans une image ou une série d'images d'une

même scène. Nous présenterons également dans le premier chapitre des problèmes similaires, afin de mieux situer notre travail par rapport à d'autres recherches effectuées dans le domaine du traitement d'images couleur.

Dans le deuxième chapitre, nous présenterons les notions essentielles à la compréhension du problème de la localisation couleur. Ainsi, nous nous attarderons sur le concept d'histogramme, concept central des algorithmes étudiés dans ce mémoire. De plus, nous étudierons en détail plusieurs algorithmes, dont le CHS, qui permettent de localiser un objet de référence dans une image.

La suite de ce document abordera différents aspects de l'algorithme du CHS qui ne sont actuellement pas satisfaisants. En effet, bien que les méthodes de localisation couleur aient de nombreux avantages, elles présentent des faiblesses auxquelles il faut remédier.

Le chapitre 3 détaillera les différents modes de représentation de la couleur. Nous y présenterons divers espaces, tels le RGB et le HSV, que nous classifierons selon les propriétés qu'ils possèdent. Nous terminerons ce chapitre par une étude comparative qui nous permettra d'identifier les espaces de couleur qui sont le plus adaptés à notre problème et ses contraintes.

Puis, nous nous intéresserons au problème de l'illumination des images analysées. Ce problème est très souvent abordé dans la littérature car il s'applique à toute méthode basée sur des informations couleur. En effet, comme nous le verrons, la perception des couleurs est fortement modifiée selon l'éclairage de la scène où l'on veut localiser un objet. La méthode du CHS peut être perturbée par des changements d'éclairage et nous étudierons plusieurs possibilités d'amélioration afin de pallier ce problème.

Lors de déplacements, la taille de la cible dans la scène peut varier beaucoup dans une série d'images successives. Le CHS est plutôt tolérant à de faibles variations de taille, mais il lui manque une capacité d'adaptation plus conséquente. Le chapitre 5 étudiera cet aspect et proposera une solution, la méthode du CHF.

Enfin, nous présenterons le problème de l'évaluation de la qualité des solutions du CHS. Effectivement, comme nous le verrons plus loin, le CHS détermine toujours une position pour la cible mais, malheureusement, l'algorithme peut trouver une position erronée. Il est donc nécessaire de pouvoir

disposer de critères quantitatifs pour déterminer la validité d'une solution de localisation.

Nous terminerons ce mémoire par une conclusion sur les travaux que nous avons réalisés lors de notre stage. Nous y évaluerons le travail effectué durant ces cinq mois et proposerons des perspectives de recherche pour le laboratoire TROP.

Problématique

Chapitre 1

Contexte de la problématique

1.1 Introduction

Dans ce chapitre, nous allons d'abord présenter brièvement le groupe TROP qui nous a accueillis durant le stage qui a conduit à ce mémoire. Il est constitué surtout de professeurs, maîtres de conférence et étudiants thésards. Ils étudient principalement les réseaux de neurones, mais nous verrons qu'ils ne se limitent pas à ce domaine. Nous expliquerons également comment nous avons procédé lors de nos recherches et travaux à Mulhouse.

Ensuite, nous détaillerons le problème de l'asservissement visuel d'un bras robotique. Il s'agit en effet de notre problématique de départ. Pour un certain nombre de raisons, nous avons restreint nos travaux à une sous-partie de ce sujet : la localisation d'une cible dans une image couleur. Nous n'avons donc pas tenu compte des aspects matériels qui sont notamment le pilotage et le déplacement du robot. Ainsi, nous expliquerons les différentes facettes de ce sujet, de même que les contraintes que le groupe TROP s'est fixées dans le cadre de ses recherches.

Nous présenterons également quelques problèmes similaires à celui de la détection d'une cible dans une image. En effet, en recherchant parmi les articles et ouvrages scientifiques dont le sujet fait partie de ce domaine, nous avons trouvé un certain nombre de matières qui revenaient fréquemment. Dans ces derniers, les buts poursuivis étaient différents, mais le même type de techniques étaient employées. Dès lors, nous avons trouvé intéressant de les présenter.

1.2 Environnement de travail du groupe TROP

Afin de pouvoir réaliser ce mémoire, nous avons été accueillis par le groupe TROP[1]. Ce dernier fait partie du laboratoire MIPS[2] des Facultés des Sciences et Techniques de l'université de Haute-Alsace à Mulhouse, en France.

Les travaux du groupe TROP concernent principalement l'intégration de la vision dans la commande robotique. Dans le cadre de leurs recherches, ils emploient une plate-forme robotique composée d'un système de vision, d'un bras robotique et d'un ensemble d'ordinateurs. L'utilisation de la vision nécessite la résolution de plusieurs problèmes :

- reconnaissance de l'objet à saisir,
- localisation de cet objet dans la scène,
- calcul du mouvement du robot.

Le premier problème est résolu très facilement. Le système de vision fournit des images de la scène. Pour définir l'objet à saisir, il suffit de sélectionner une partie d'une image.

Leurs recherches dans le domaine de la vision couleur ont pour but de reconnaître et localiser un objet dans une image sans utiliser de marqueurs[3]. C'est dans ce contexte que s'est situé notre stage au sein du laboratoire TROP.

Le groupe TROP a déjà étudié avec plusieurs thésards la possibilité d'utiliser les réseaux de neurones pour remplacer le calcul du mouvement du robot par un algorithme d'apprentissage.

Les recherches que nous avons effectuées lors de notre stage étaient la prolongation d'un travail réalisé par Zimmermann [27] dans le cadre d'un DEA. Lorsque nous sommes arrivés, une application de test, programmée en C++, avait été mise en chantier. Nous avons contribué à son développement tout au long de notre séjour à Mulhouse. Ce programme pouvait utiliser comme source, aussi bien des séquences pré-enregistrées que des images fournies par une « webcam », afin d'y appliquer un algorithme (tels que ceux expliqués

[1] *Contrôle Neuromimétique et Robotique* (`http://www.trop.uha.fr`)

[2] *Modélisation, Intelligence, Processus, Système* (`http://www.mips.crespim.uha.fr`)

[3] Un marqueur est, par exemple, un bout de papier, dont la couleur se distingue du reste de la scène, que l'on place sur la cible. L'utilisation de marqueurs facilite la localisation d'une cible.

dans les sections 2.5.1 et 2.5.2) et visualiser les résultats en temps réel. Ainsi, lorsque nous travaillions sur un algorithme, nous procédions en deux étapes :

- élaboration de l'algorithme et tests sur images fixes sous Matlab 7,
- test de l'algorithme en temps réel, amélioration et optimisation sous l'application C++.

L'avantage d'un tel procédé réside dans la facilité que nous donne le logiciel Matlab à visualiser tout résultat intermédiaire. Au contraire, en C++, l'affichage graphique de certaines variables est assez long à mettre en oeuvre. De plus, Matlab met à notre disposition une quantité importante de fonctions mathématiques et de traitement d'images. Ainsi, nous pouvons aisément développer et tester un algorithme de traitement d'images. Néanmoins, toutes ces facilités ont un coût, principalement en termes de temps de calcul. C'est pourquoi, la deuxième étape consiste à transposer le code dans l'application C++ et l'optimiser pour le temps réel.

Les algorithmes que nous allons présenter dans le chapitre 2 ont d'ailleurs été presque tous implémentés dans cette application afin de comparer facilement leurs résultats. En effet, ce programme a été doté de fonctionnalités facilitant la sauvegarde d'informations à des fins d'interprétation et de réutilisation ultérieure. Une capture d'écran de la version finale du programme est disponible en annexe E, accompagnée d'une brève description de ses fonctionnalités.

1.3 Description de la problématique

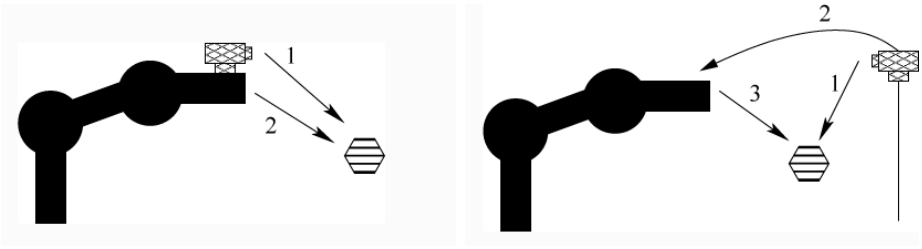
Comme nous l'avons déjà dit, le problème que nous avons étudié est celui de l'asservissement visuel d'un bras robotique. Le but était de déplacer ce dernier vers un objet désigné comme cible. La résolution d'un tel problème peut mener à de nombreuses applications pratiques dont, par exemple, la possibilité pour un handicapé, immobilisé dans une chaise roulante équipée d'un bras robotique et d'une caméra, de saisir une tasse de café.

L'asservissement visuel d'un robot dans le but de saisir un objet distant est un processus qui peut se résumer en quatre étapes :

1. définition de l'objet à récupérer,
2. détermination de l'emplacement de l'objet,
3. mouvement du robot en direction de l'endroit localisé,

4. retour à la deuxième étape.

La configuration envisagée ici est une configuration « eye-in-hand », c'est-à-dire où la caméra est fixée à l'extrémité du bras robotique. Effectivement, l'autre approche possible — une caméra immobilisée à un endroit précis — limite les possibilités d'application. Les deux situations sont illustrées sur la figure 1.1.



(a) « eye-in-and » (b) Caméra fixe

FIG. 1.1 – Configurations de caméra possibles

Il est tout à fait possible de fixer une caméra sur le bras robotique disponible au laboratoire TROP. Néanmoins, il faut valider soigneusement un algorithme de vision avant de penser à l'intégrer dans une solution robotique. Comme il s'agit de méthodes innovantes, des tests intensifs sont nécessaires pour que tous les cas particuliers (détections incorrectes, cible hors du champ de vision, ...) soient gérés. En effet, les risques expérimentaux sont relativement élevés et un algorithme défectueux pourrait mal détecter la présence de l'objet-cible dans la scène, ordonner au robot d'effectuer des mouvements qui lui sont impossibles et y causer des dégâts matériels. Nous avons donc choisi de procéder autrement.

Nous avons envisagé le problème sous un autre angle : plutôt que déplacer le robot (et la caméra) et repérer l'objet tout au long des mouvements, nous proposons de suivre la trajectoire d'un objet mobile. Cela permet de faire abstraction des aspects mécaniques en jeu (la cible étant déplacée manuellement).

Ces deux problèmes sont similaires, mais diffèrent cependant sur un point. Dans les deux cas, la cible est mobile tout au long d'une séquence d'images

mais le déplacement du robot entraîne une modification du reste de l'image (l'arrière-plan) qui n'apparaît pas dans la démarche proposée ici.

Nous n'étudierons donc que les deux premières étapes de l'asservissement visuel : la définition d'une cible et sa localisation dans une image.

Les deux contraintes principales que le groupe TROP s'est fixées sont les suivantes :

– calculs en temps réel,
– faibles coûts de matériel.

La première est la condition *sine qua non* pour que les résultats du projet puissent être exploitables. Effectivement, certains algorithmes ne sont pas entièrement satisfaisants pour une utilisation en temps réel. Par exemple, Cubber et al. [12] ont développé un ensemble d'algorithmes pour le suivi d'une cible, mais les meilleurs résultats nécessitaient un temps de calcul de l'ordre de 100 ms. En outre, peu d'auteurs évoquent réellement le temps de calcul de leur solution, ils se contentent d'en donner une vague idée sans même préciser le matériel utilisé. L'objectif du TROP étant de permettre à un robot de saisir un objet, le temps de détection de la cible dans l'image ne devrait pas dépasser 40 ms (la caméra capture 25 images par seconde).

La seconde restriction concerne les coûts. En effet, certaines publications, telles que celle de Chang et Krumm [10], citent d'excellents résultats lors d'expériences similaires aux nôtres, mais l'usage de matériel plus onéreux et d'hypothèses restrictives leur permet de maîtriser de nombreux paramètres, leur facilitant ainsi le travail. Par exemple, ils faisaient usage de lentilles spécifiques, la caméra était configurée précisément et les modèles étaient définis sur fond noir. Au contraire, le groupe TROP souhaite employer du matériel standard (ordinateur de bureau moyen et « webcam ») afin de permettre une grande diffusion à la solution. Ainsi, en prenant l'exemple de l'handicapé voulant prendre une tasse de café sur une table, cela permettrait à cette personne d'obtenir un système pratique à un prix abordable.

1.4 Problèmes similaires

1.4.1 Introduction

Dans la plupart des documents relatifs au traitement d'images couleur, on retrouve les mêmes applications envisagées. De plus, certains se focalisent

sur leur sujet spécifique et conçoivent des algorithmes qui ne se révèlent fonctionnels que dans leur cas particulier d'étude. Il s'agit donc d'une philosophie de travail opposée à celle du TROP, qui veut rester général et ne pas imposer de contraintes à la scène filmée.

Dans cette section, nous proposons de passer en revue trois exemples similaires à la localisation d'une cible dans une image couleur :

- suivi de joueurs dans les matches de football,
- recherche d'une image dans une base de données,
- suivi de visages.

1.4.2 Suivi de joueurs dans les matches de football

De part sa médiatisation, le football implique des enjeux économiques très importants. En effet, les salaires des joueurs sont colossaux et les sponsors investissent beaucoup d'argent dans leurs clubs. Dès lors, l'analyse du déroulement d'un match est une étape clé dans la vie d'un club : il est crucial de comprendre chaque seconde du match afin d'améliorer les résultats.

Du côté des média, le problème est le même. Les chaînes télévisées dépensent des montants importants pour obtenir les droits de retransmission des matches. Pour satisfaire les téléspectateurs, il leur faut donc fournir un service de qualité au niveau de l'analyse en temps réel des actions des joueurs et montrer des statistiques pertinentes.

C'est pourquoi l'analyse automatique de séquences d'un match est très intéressante à étudier. Ce problème a notamment été abordé en profondeur par Vandenbroucke [25] dans le cadre d'une thèse de doctorat.

Il est évident que ce problème est très similaire à celui qui nous préoccupe. En effet, il faut détecter, sur chaque image d'une séquence, la position des joueurs et du ballon et cela, en temps réel. Evidemment, l'analyse est relativement plus facile à faire, car les hypothèses faites sur la scène à décortiquer sont très fortes : terrain vert, deux couleurs de maillots différentes, ... La présence du public, des affiches et logos de sponsors complique cependant la tâche.

1.4.3 Recherche d'une image dans une base de données

La recherche d'images dans une base de données (« image retrieval ») est un domaine de recherche de plus en plus prisé. Effectivement, les capa-

cités de stockage ne cessant de grandir, les disques durs se trouvent envahis de données multimédia telles que des photos et des fichiers musicaux. Des programmes de catalogues musicaux permettent de retrouver aisément une chanson particulière grâce aux données encodées dans ces fichiers (titre, chanteur, ...), mais les images ne sont accompagnées d'aucune description, si ce n'est le nom du fichier. D'ailleurs, ce problème est présent partout, mais il prend une plus grande importance dans certaines institutions.

Bon nombre de scientifiques travaillent activement à développer des techniques permettant de récupérer des images similaires à une image donnée dans une base de données pouvant contenir parfois plus de 10000 fichiers. En fournissant en entrée une photo d'une voiture rouge par exemple, on désire obtenir toutes les photos sur lesquelles il y a une voiture rouge.

Ce sujet a été étudié entre autres par Pass et Zabih [18] et Pecenovic et al. [19]. Un mémoire a d'ailleurs déjà été réalisé sur ce sujet aux Facultés Universitaires Notre-Dame de la Paix de Namur [2]. La taille considérable de ces bases de données impose aux chercheurs des contraintes de temps de calcul proches des nôtres. Il y a cependant une nuance importante entre ce sujet et l'objet de ce mémoire : le but d'une recherche dans une base de données consiste à retrouver des similarités et non un objet précis. Ce point implique souvent l'utilisation de méthodes plus lentes que celles que nous allons employer.

1.4.4 Suivi de visages

Ces dernières années, une quantité importante de méthodes basées sur l'analyse des couleurs ont été mises au point. Une partie de ces dernières se sont focalisées sur un type bien particulier de suivi.

Ainsi, le suivi de visages est un sujet largement traité dans la littérature :
- méthodes statistiques de Bernier et al. [3],
- méthodes des gradients d'intensités de Birchfield [4],
- CamShift de Bradski [7].

Il en existe d'autres puisque le cas du suivi de visages humains est un problème qui intéresse beaucoup de personnes. En effet, dans une société moderne où la sécurité prend de plus en plus d'importance, il serait intéressant de doter les caméras de surveillance de la capacité de suivre le mouvement du visage. Ainsi, plutôt que d'effectuer sans cesse les mêmes mouvements de

rotation autour d'un axe, elles s'adapteraient aux situations auxquelles elles sont confrontées.

Ces techniques reposent sur la connaissance de la dynamique du système et formulent donc des hypothèses sur les éléments suivants :

- forme de l'objet,
- rapidité de mouvement,
- uniformité de la distribution des couleurs.

Dès lors, ces hypothèses facilitent le travail des chercheurs et ne sont pas facilement adaptables à notre cas. Leurs hypothèses sont effectivement beaucoup plus fortes que les nôtres.

1.5 Conclusion

Dans ce chapitre, nous avons décrit l'environnement de travail du groupe TROP, à Mulhouse. Nous avons également présenté la problématique de départ : l'asservissement visuel afin de saisir des objets. Il s'agit du problème de diriger un robot en se basant sur des informations visuelles.

Au cours de notre stage, nous avons uniquement étudié le problème de la localisation d'une cible sur une image, sans tenir compte des aspects robotiques de la question. Certaines contraintes étaient imposées, à savoir utiliser du matériel bon marché et permettre une utilisation en temps réel.

Enfin, nous avons présenté quelques problèmes semblables au nôtre, tels que le suivi de joueurs de football au cours d'un match et la recherche d'images dans des bases de données multimédia.

Chapitre 2

Localisation couleur

2.1 Introduction

Le problème principal que nous étudions est donc celui de la détection d'une cible prédéfinie dans une scène. Les images de la scène sont capturées au moyen d'une caméra couleur, il faut ensuite les analyser pour y localiser la cible. Dans ce chapitre, nous allons présenter les concepts fondamentaux liés à l'utilisation d'images en couleurs, nous présenterons également quelques techniques de localisation couleur.

Il s'agit d'un problème relativement complexe pour lequel le monde de la recherche est assez actif. Il est donc impossible d'en faire un état de l'art complet. C'est pourquoi nous n'étudierons en détail que les techniques qui nous intéressent, tandis que les autres seront seulement citées. En effet, puisque certains scientifiques ne travaillent pas avec les contraintes que nous nous sommes fixées, leurs résultats, bien qu'intéressants, ne sont pas applicables à notre sujet.

De nombreuses méthodes ont été proposées afin de résoudre le problème de la localisation d'une cible de référence dans une image. Jusqu'à récemment, la principale approche utilisée se basait sur une information géométrique de l'objet cherché. La couleur était en effet considérée comme secondaire dans le processus de reconnaissance. Ainsi, ce qui définit une chaise, c'est sa forme et non sa couleur. Une chaise peut changer de couleur et de texture mais reste une chaise.

Cette constatation, bien que fondée, traduit mal les contraintes posées par beaucoup de problèmes de localisation dans une image. Il n'est en ef-

fet pas toujours nécessaire de devoir retrouver « une » chaise mais bien une chaise en particulier, celle-ci ayant une combinaison de couleurs bien définie. De plus, certains objets sont différenciés grâce à leurs couleurs. Par exemple, les boîtes de jus d'orange ont souvent une forme similaire mais se distinguent par leurs inscriptions. Les couleurs d'un objet en particulier sont donc généralement de très bons indicateurs à utiliser afin de les identifier.

Nous pouvons aussi remarquer que, d'un point de vue plus technique, les informations sur la forme d'un objet dans une image sont très sensibles à la résolution, elles changent radicalement selon le point de vue et demandent un travail important pour leur extraction (et donc du temps). Tous ces inconvénients rendent l'analyse des formes beaucoup plus complexe à utiliser dans le cas de la localisation d'objets bien définis.

Lors de notre présentation des techniques de localisation couleur existantes, nous aborderons d'abord quelques notions importantes liées à la couleur.

Nous présenterons ensuite le concept de « signature couleur ». C'est en effet sur cette notion que se basent tous les algorithmes existants. Nous nous attarderons également sur un type de signature particulier, les histogrammes couleur, et nous introduirons quelques mesures de distance. Ces dernières sont nécessaires pour pouvoir estimer le degré de similarité entre deux images.

Nous étudierons en détail quelques techniques de localisation : la « back-projection » (et ses variantes), le CHS et le corrélogramme couleur.

Le principe général employé dans les algorithmes que nous allons présenter est le suivant : la cible est définie en termes de pixels colorés et il s'agit de la retrouver dans une autre image en effectuant des comparaisons. A chaque méthode de comparaison différente est associé un algorithme différent.

2.2 Notions relatives à la couleur

2.2.1 Couleur

A la base de toute perception de la couleur chez l'être humain, il y a une source de lumière. En effet, lorsqu'il fait noir, nous sommes incapables de distinguer les couleurs des objets qui nous entourent.

Grâce à Isaac Newton, nous savons depuis 1666 que la lumière peut être séparée en rayons lumineux monochromatiques. Chacun de ces rayons est constitué de radiations de même longueur d'onde. Autrement dit, la longueur d'onde d'un rayon en définit la couleur. Seules celles comprises entre 400 et 700 nanomètres peuvent être détectées par le système visuel humain ; elles sont affichées sur la figure 2.1.

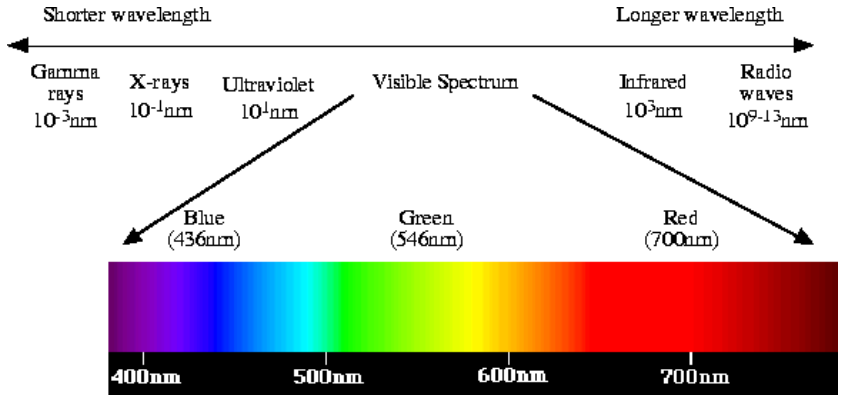


FIG. 2.1 – Longueurs d'onde visibles par l'oeil humain (source : [16])

L'étude des couleurs se fonde surtout sur la théorie trichromatique. Toute couleur peut être reproduite de façon identique par le mélange algébrique, en proportions définies de manière unique, de trois couleurs dites primaires : c'est le principe de trivariance visuelle. Le choix de ces trois couleurs est arbitraire à une seule condition : chacune d'entre elles ne doit pas pouvoir être reproduite par un mélange des deux autres. Cette théorie repose sur des expériences qui ont été menées dans des conditions expérimentales bien particulières. Elles consistaient, pour un observateur quelconque, à comparer un

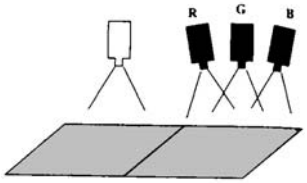


FIG. 2.2 – Expérience d'égalisation (source : [24])

stimulus visuel (3 lumières monochromatiques, émises en certaines quantités) à un stimulus de référence. Cette expérience, appelée « expérience d'égalisation », est schématisée sur la figure 2.2 (avec le rouge, vert et bleu comme

primaires).

En résumé, pour trois primaires P_1, P_2 et P_3 et une couleur C_λ, l'expérience permettait de calculer les coefficients p_1, p_2 et p_3 tels que

$$C_\lambda = p_1(\lambda)\,[P_1] + p_2(\lambda)\,[P_2] + p_3(\lambda)\,[P_3] \tag{2.1}$$

2.2.2 Teinte

La teinte, également appelée tonalité chromatique, correspond à la longueur d'onde prédominante dans la couleur. Selon la définition de l'AFNOR[1], il s'agit de *l'attribut de la sensation visuelle qui a suscité des dénominations de couleur telles que bleu, vert, jaune, ...*

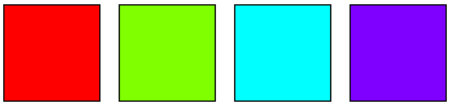


FIG. 2.3 – Couleurs ayant différentes teintes

La figure 2.3 donne une explication plus visuelle du concept de teinte : des variations de teinte sont appliquées sur une même couleur (la saturation et la luminosité restent donc identiques).

2.2.3 Saturation

La saturation caractérise l'aspect vif ou pâle d'une couleur. Plus elle est grande et plus la couleur d'onde prédominante est mélangée à du blanc. Selon la définition de l'AFNOR, c'est *l'attribut de la sensation visuelle qui permet d'estimer la proportion de couleur chromatiquement pure contenue dans la sensation totale.* En d'autres termes, la saturation indique le niveau de pureté de la couleur perçue. Par exemple, le rouge vif est une couleur saturée alors que le rose pâle est une couleur désaturée.

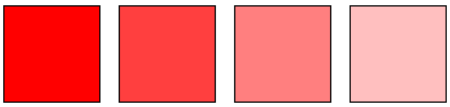


FIG. 2.4 – Couleurs ayant différentes saturations

[1] *Association Française de Normalisation* (`http://www.afnor.fr`)

La figure 2.4 donne une explication plus visuelle du concept de saturation : des variations de saturation sont appliquées sur une même couleur (la teinte et la luminosité restent donc identiques).

2.2.4 Luminosité

La luminosité est une perception que nous traduisons par des termes tels que « clair », « foncé », « lumineux » ou encore « sombre ». C'est, d'après l'AFNOR, *l'attribut de la sensation visuelle selon lequelle une surface paraît émettre plus ou moins de lumière.* Dans la littérature, elle est aussi appelée « intensité » ou « luminance ».

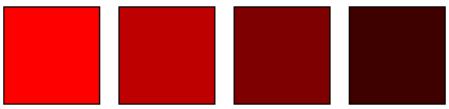


FIG. 2.5 – Couleurs ayant différentes luminosités

La figure 2.5 donne une explication plus visuelle du concept de luminosité : des variations de luminosité sont appliquées sur une même couleur (la teinte et la saturation restent donc identiques).

2.2.5 Chrominance

La chrominance désigne la partie des informations couleur qui n'inclut pas la luminance, à savoir la teinte et la saturation. Il s'agit donc principalement de la couleur primaire à laquelle une couleur correspond, sans prendre en compte les faibles variations (bleu foncé, bleu clair, ...). En d'autres termes, telle que définie par Boutell [5], *la chrominance est une mesure objective de la couleur d'un objet, hormis toute considération de luminosité.*

2.2.6 Illuminant

Il existe de nombreuses sources lumineuses : le soleil bien sûr, mais aussi les flammes ou encore les différents types de lampes existants (lampes halogènes, tubes néons, ...). De plus, la lumière du soleil varie selon certains facteurs tels que l'heure du jour et les conditions atmosphériques. C'est pour cela que la CIE[2] a introduit la notion d'illuminant.

[2] *Commission Internationale de l'Eclairage* (`http://www.cie.co.at`)

Catégorie	Description de l'illuminant
A	Il correspond à la source lumineuse produite par une lampe à filament de tungstène de 500W.
B	Il correspond à la lumière directe du soleil à midi.
C	Il correspond à une source lumineuse dont le niveau d'illumination est équivalent à celui du ciel.
D	Il correspond à différentes lumières du jour. Il en existe plusieurs dont les plus utilisés sont D_{50}, D_{55}, D_{65} et D_{75}. Ils remplacent les illuminants B et C, car ils représentent plus précisément la lumière du jour.
E	Il correspond à une source de référence normalisée (blanc équi-énergétique). Il ne correspond à aucune source lumineuse réelle, son intérêt est donc purement théorique.
F	Il correspond à la lumière émise par une lampe fluorescente. Il en existe douze différents numérotés de F1 à F12.

TAB. 2.1 – Illuminants standards

Certaines sources de lumière correspondent à des conditions d'observation assez courantes. Ces dernières ont été normalisées par la CIE qui les a appelées « illuminants ». Le tableau 2.1 contient les principales catégories d'illuminants.

Cette description des illuminants explique à quoi chacun d'eux correspond. Pour les mesurer, il faut se référer à leur température de couleur, c'est-à-dire à la température à laquelle il faudrait porter un corps noir pour obtenir une répartition spectrale d'énergie identique à celle de la source. Néanmoins, cette définition dépasse le cadre de ce document et ne sera pas détaillée ici. Une présentation plus complète des illuminants peut être trouvée, par exemple, dans Vandenbroucke [25].

2.2.7 Blanc de référence

Le blanc de référence est une couleur prise en compte lors de certaines manipulations de couleurs. Il est déterminé par l'illuminant choisi et il influence les transformations entre espaces de représentation de la couleur. Ces dernières seront expliquées dans le chapitre 3.

2.2.8 Gamma

Généralement, la luminance générée par un appareil physique n'est pas une fonction linéaire du signal vidéo. Pour reproduire correctement la luminance, il est nécessaire de compenser cette non-linéarité. C'est cette compensation que l'on appelle « correction gamma ». Dans le cas des moniteurs CRT (les écrans à tube cathodique), la luminance produite par l'écran est proportionnelle à la tension appliquée lorsqu'elle est élevée à la puissance $\gamma = 2.5$.

2.3 Signatures d'images couleur

La recherche en traitement d'images s'est focalisée pendant longtemps sur les images en noir et blanc ou, au mieux, en niveaux de gris (256 niveaux différents). Ces dernières années, une tendance nouvelle est apparue : l'utilisation d'images en couleurs. Les méthodes ayant été développées auparavant ne peuvent cependant pas s'appliquer telles quelles aux images couleur, car la quantité d'informations à gérer est beaucoup plus importante.

Utiliser les images couleur sans traitement préalable ne saurait être envisagé pour cette étude. Ainsi, une comparaison de deux images pixel par pixel n'est pas réaliste pour une application en temps réel à cause des dimensions des images employées[3]. De plus, une telle méthode de comparaison ne permet pas de tenir compte des variations de taille, d'orientation, ... Il faut donc trouver une autre technique.

Il est proposé d'indexer les images avant de les traiter, en se basant sur des caractéristiques de bas niveau telles que la couleur, la texture, ... Cette signature doit être compacte, mais doit néanmoins être suffisamment complète pour bien caractériser l'image. Une ou plusieurs métriques de similarité doivent également être déterminées par rapport à cette signature.

Pour un relevé plus complet et plus détaillé des signatures qu'il est possible d'employer, on peut se référer notamment à Tremeau et al. [24].

L'attribut principalement utilisé lors de l'élaboration d'une signature est, sans surprise, la couleur. Le nombre de couleurs existantes est cependant trop élevé, c'est pourquoi on applique généralement aux images un traitement

[3] 320 pixels de largeur et 240 pixels de hauteur, pour un total de 76800 pixels, ce qui représente une trop grande quantité de données à traiter.

préliminaire avant d'en calculer la signature (voir section 2.4). Voici quelques signatures basées sur la couleur :

- histogrammes,
- moments de la distribution couleur,
- histogrammes cumulés.

L'histogramme est la signature la plus utilisée dans le monde du traitement d'images couleur, il s'agit d'un vecteur $(h_{c_1}, \ldots, h_{c_n})$ contenant le nombre d'occurences de chaque couleur c_i dans l'image. Sa taille dépend donc du nombre n de couleurs présentes dans cette dernière. Cette signature est utilisée dans la technique que nous allons étudier en détail dans ce document, nous y reviendrons plus en détail dans la section 2.4.

Les moments sont utilisés lorsque la taille de l'index est un facteur critique. La signature se limite alors aux premiers moments de la distribution des couleurs. Voici la définition des trois premiers moments pour chaque plan i[4] [24] :

$$E_i = \frac{1}{N} \sum_{j=1}^{N} p_{ij} \tag{2.2}$$

$$\sigma_i = \frac{1}{N} \sqrt{\left(\sum_{j=1}^{N} (p_{ij} - E_i)^2 \right)} \tag{2.3}$$

$$s_i = \frac{1}{N} \sqrt[3]{\left(\sum_{j=1}^{N} (p_{ij} - E_i)^3 \right)} \tag{2.4}$$

où N est le nombre de pixels dans l'image et p_{ij} est la $i^{ème}$ composante de la couleur du pixel j.

Les histogrammes cumulés sont utilisés afin d'obtenir plus de robustesse vis-à-vis de l'espace couleur utilisé. Ils nécessitent un ordonnancement réfléchi des couleurs c_i, ce qui est assez lourd en termes de temps de calcul et c'est pourquoi nous ne nous attarderons pas dessus. Ils sont définis comme suit [24] :

$$\tilde{h_{c_i}} = \sum_{c_j \leq c_i} h_{c_j} \tag{2.5}$$

[4] Chaque plan i contient les coefficients p_i des couleurs des pixels de l'image (voir section 2.2).

où $c_j \leq c_i$ impose l'ordonnancement des couleurs.

D'autres types de signatures sont les signatures mélangeant informations de texture (corrélations spatiales locales) et de couleur. On peut citer par exemple le corrélogramme couleur, qui sera détaillé dans la section 2.5.3.

Enfin, il y a aussi des signatures utilisant la couleur et des données spatiales dont, notamment, le vecteur de cohérence couleur. La notion de cohérence dépend de l'appartenance d'un pixel à une région et une région est définie par un ensemble de pixels voisins de couleur identique. Il s'agit donc de répartir les pixels en zones de couleur[5].

Nous n'étudierons pas plus en détail ces signatures car l'exploitation d'informations de texture ou d'espace implique un plus grand temps de calcul, car des informations sur les pixels voisins sont utilisées alors que l'histogramme, comme expliqué dans la section suivante, ne nécessite qu'une seule lecture de chaque pixel. Dès lors, il semble plus avantageux de favoriser les histogrammes aux signatures mixtes dans le cadre d'une application en temps réel. Nous nous pencherons cependant sur le cas du corrélogramme couleur dans la section 2.5.3. Nous y verrons que l'utilisation d'une signature plus complexe n'implique pas nécessairement une amélioration des résultats.

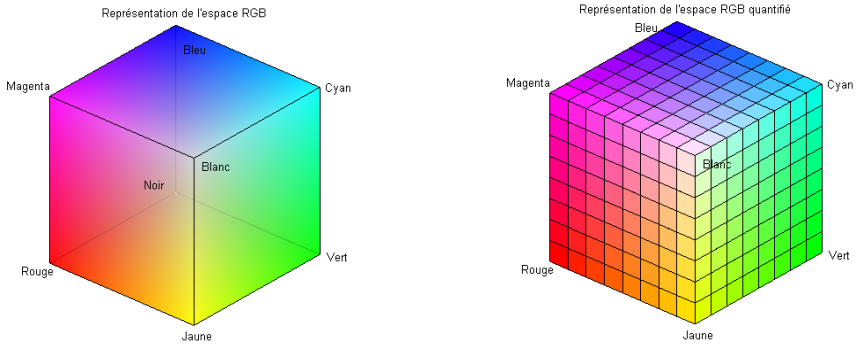
2.4 Quantification et histogramme

2.4.1 Quantification

L'histogramme d'une image est un vecteur contenant, pour chaque couleur possible, le nombre de pixels de cette couleur dans l'image. Il est donc nécessaire de réduire l'ensemble des couleurs possibles car, si on emploie des images codées sur 24 bits, l'histogramme serait un vecteur contenant plus de 16 millions d'éléments. Il n'y aurait ainsi pratiquement aucun avantage à travailler avec les histogrammes puisqu'ils requerraient plus d'espace mémoire que l'image elle-même.

Dès lors, avant de calculer l'histogramme d'une image, il est nécessaire d'effectuer un traitement appelé « quantification » ou « indexation ». Par cette opération, on modifie l'espace des couleurs afin d'en réduire la taille. Concrètement, il s'agit de discrétiser l'espace des couleurs.

[5] Pour plus de détails, se référer à Tremeau et al. [24].



(a) Espace RGB (b) Espace RGB quantifié

FIG. 2.6 – Représentation graphique de l'espace RGB

Considérons par exemple l'espace RGB, affiché sur la figure 2.6 (a). Les trois axes correspondent aux axes rouge, vert et bleu. Une quantification possible de cet espace est présentée sur cette même figure (b) : chaque composante peut prendre 8 valeurs différentes, ce qui donne un total de 512 couleurs.

Ainsi, chaque axe du cube RGB a été compartimenté, ce qui a donné lieu à un certain nombre de sous-cubes SC_k. A chaque SC_k, une couleur COL_k est assignée (celle du centre de SC_k par exemple). Ensuite, on analyse la couleur $P_{i,j}$[6] de chaque pixel de l'image et on détermine à quel sous-cube elle appartient. Soit SC_m ce sous-cube. Il suffit alors de remplacer $P_{i,j}$ par COL_m. Dans la pratique, l'ensemble des COL_k est stocké dans un vecteur appelé palette et les $P_{i,j}$ sont remplacés par l'index de la couleur dans ce vecteur. Dans la littérature, les sous-cubes — ou plus généralement, les sous-espaces — sont appelés « bins ».

Sur la figure 2.6 (b), chaque composante peut prendre 8 valeurs différentes, elles sont donc codées chacune sur 3 bits. Dans la suite de ce document, nous nous référerons à une telle quantification par la notation « 3-3-3 » ou encore « 333 ». En pratique, nous utiliserons des quantifications définies de sorte que la somme du nombre de bits assignés à chaque composante soit 8 (1 octet). Pour certaines techniques plus avancées, un nombre plus important de bits est nécessaire, mais cela reste rare.

[6] $P_{i,j}$ est la couleur du pixel situé en (i,j).

Il est important de noter que ce qui a été présenté ci-dessus fait référence à une quantification dite « uniforme », car l'espace a été divisé en sous-espaces de même taille. Il existe des quantifications non-uniformes qui prennent en compte la distribution réelle des couleurs afin de réduire la distorsion créée lors de la diminution du nombre de couleurs. Par exemple, si une image représente un paysage enneigé, pour garder une bonne qualité, il peut être envisageable d'utiliser des sous-espaces plus fins autour de la couleur blanche et des sous-espaces plus grossiers ailleurs. En termes de « bins », il s'agit de ne pas tenir compte des « bins » peu remplis et de subdiviser les autres (la figure 2.7 montre un exemple pour lequel certains « bins » sont vides alors que d'autres sont bien remplis). Cependant, de telles quantifications nécessitent plus de temps de calcul.

La quantification non-uniforme est également utilisée pour la compression de données : une représentation optimale de l'image est recherchée, pour une taille fixée. Une telle opération nécessite plus de temps de calcul car une connaissance globale de l'image est requise. De plus, une quantification non-uniforme complique la comparaison entre deux images. En effet, en considérant F et G, les quantifications appliquées à deux images différentes ($F \neq G$ dans la majorité des cas), pour une couleur c, on a que $F(c) \neq G(c)$. L'utilisation de quantification non-uniforme complique donc la comparaison des couleurs.

2.4.2 Histogramme

Comme nous l'avons déjà brièvement expliqué, l'histogramme contient le nombre d'occurrences de chaque couleur dans l'image. Nous pouvons donc le définir comme étant la distribution des densités de couleur dans l'image. Plus formellement, si l'image est quantifiée de sorte que l'espace couleur soit réduit à n classes couleur $(c_1, \cdots, c_n)$, alors l'histogramme de l'image est un vecteur à n composantes, noté $(h_{c_1}, \cdots, h_{c_n})$. Chaque h_{c_i} contient le nombre de pixels dans l'image ayant pour couleur c_i. On a $\sum_{i=1}^{n} h_{c_i} = (largeur \times hauteur)$.

La figure 2.7 montre une image indexée et son histogramme. Pour représenter ce dernier, chaque bâtonnet a été dessiné en utilisant la couleur à laquelle il correspondait.

L'avantage des histogrammes est qu'ils sont invariants à la géométrie des objets. Effectivement, ils représentent la distribution des couleurs, mais ne

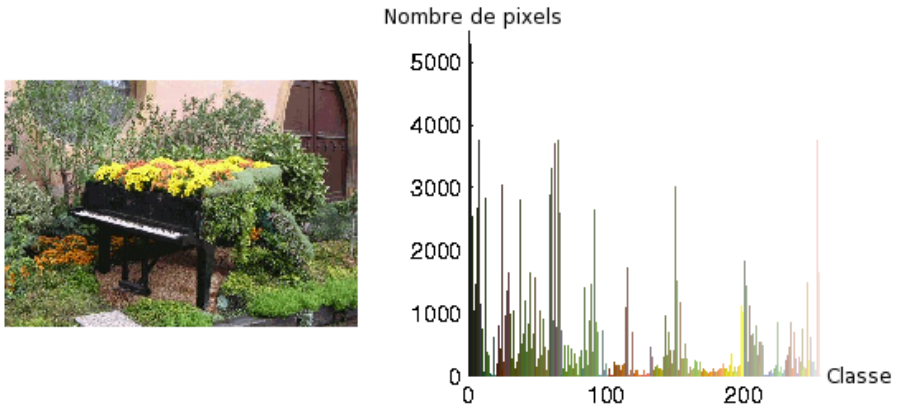


FIG. 2.7 – Image indexée (256 couleurs) et son histogramme

donnent aucun renseignement au sujet de la répartition spatiale de celles-ci. Ainsi, le déplacement ou la rotation d'un objet dans une scène ne changera pas ou peu l'histogramme de l'image. Ils sont donc très intéressants pour le sujet qui nous occupe. Il y a néanmoins un inconvénient : un changement d'illumination de la scène entraîne une translation de l'histogramme qui fausse la reconnaissance de l'objet-cible. Ce problème sera abordé en détail plus tard dans ce document (voir chapitre 4).

FIG. 2.8 – Deux images différentes ayant un même histogramme

Il y a un autre inconvénient, lié à l'invariance à la géométrie : deux objets ayant une même répartition de couleurs, comme sur la figure 2.8, auront un même histogramme. Ainsi, l'invariance à la géométrie est une arme à double tranchant : elle permet de tolérer des déplacements et rotations de la cible, mais elle peut aussi confondre celle-ci avec un autre objet.

2.4.3 Mesures de distance

Après avoir choisi la signature à utiliser, il reste à déterminer une mesure de distance. En effet, notre but étant de localiser une cible dans une image, il faut pouvoir déterminer à quel point deux images de même taille sont (dis)semblables.

Dans notre cas, puisque l'histogramme est un vecteur à n composantes (n étant le nombre de couleurs), il est possible d'utiliser les distances de Minkowski L_1, L_p et L_∞. Voici un rappel de leur définition [24] :

$$d_{L_1}(H^1, H^2) = \sum_{i=1}^{n} |h^1_{c_i} - h^2_{c_i}| \tag{2.6}$$

$$d_{L_p}(H^1, H^2) = \left(\sum_{i=1}^{n} (h^1_{c_i} - h^2_{c_i})^p \right)^{1/p} \quad \text{où } 1 \leq p \leq \infty \tag{2.7}$$

$$d_{L_\infty}(H^1, H^2) = \max_{1 \leq i \leq n} |h^1_{c_i} - h^2_{c_i}| \tag{2.8}$$

Parmi celles-ci, L_1 est la plus employée. L'intérêt de les utiliser réside dans le fait qu'on dispose des propriétés classiques des distances (positivité, identité, symétrie et inégalité triangulaire).

Néanmoins, de nombreuses autres distances peuvent être utilisées. Parmi celles-ci, l'intersection d'histogrammes, définie par Swain et Ballard [23] est sûrement la plus populaire. La technique de localisation sur laquelle nous avons principalement travaillé (voir section 2.5.2) se base d'ailleurs sur cette distance. Elle est définie comme suit :

$$d_\cap(H^1, H^2) = \frac{\sum_{i=1}^{n} \min(h^1_{c_i}, h^2_{c_i})}{\sum_{i=1}^{n} h^1_{c_i}} \tag{2.9}$$

Cette mesure permet d'estimer le recouvrement de deux histogrammes. Il ne s'agit cependant pas d'une distance au sens mathématique, car elle ne respecte pas la propriété de symétrie. En effet, le dénominateur $\sum_{i=1}^{n} h^1_{c_i}$ correspond au nombre de pixels dans la première image. Dès lors, $d_\cap(H^1, H^2) \neq d_\cap(H^2, H^1)$, excepté lorsque les deux images comparées sont de taille identique. Pour corriger ce problème, une version modifiée a été introduite par Smith [24] :

$$d_{SMITH}(H^1, H^2) = \frac{\sum_{i=1}^{n} \min(h^1_{c_i}, h^2_{c_i})}{\min\left(\sum_{i=1}^{n} h^1_{c_i}, \sum_{i=1}^{n} h^2_{c_i} \right)} \tag{2.10}$$

On peut également citer la divergence de Kullback-Leibler, qui mesure à quel point il serait inefficace en moyenne de coder un des deux histogrammes

en utilisant l'autre comme dictionnaire. En voici la définition [24] :

$$d_{KL}(H^1, H^2) = \sum_{i=1}^{n} h_i^1 \log \frac{h_i^1}{h_i^2} \tag{2.11}$$

La divergence de Kullback-Leibler pose certains problèmes : elle n'est pas symétrique et elle est très sensible à la façon dont a été construit l'histogramme (sensible au « binning »). Jeffrey a proposé une version améliorée qui corrige ces inconvénients [24] :

$$d_J(H^1, H^2) = \sum_{i=1}^{n} \left(h_i^1 \log \frac{h_i^1}{m_i} + h_i^2 \log \frac{h_2^1}{m_i} \right) \tag{2.12}$$

où $m_i = \frac{h_i^1 + h_i^2}{2}$.

Jusqu'à ce point, les métriques que nous avons présentées considéraient toutes que les histogrammes avaient la même taille (même nombre d'éléments dans le vecteur) et que la palette était identique (la couleur du $i^{ème}$ « bin » du premier histogramme correspond à celle du $i^{ème}$ « bin » du second). Ainsi, ces calculs de distances ne tiennent compte que des similitudes « bin par bin », ce qui donne lieu à un temps de calcul réduit.

Néanmoins, d'autres métriques — des métriques « cross-bins »— ont été imaginées pour améliorer les résultats. Ces dernières ont cependant un coût que nous ne pouvons pas toujours nous permettre dans le cas qui nous préoccupe. En voici quelques unes :

- distance en forme quadratique,
- distance de Kolmogorov-Smirnov,
- EMD (« Earth Mover's Distance »).

La première utilise une matrice de similarité pour effectuer sa comparaison « inter-bins », ce qui n'est pas aisé à définir lorsqu'on ne possède aucune information sur les images. La seconde se base sur les histogrammes cummulés ($\tilde{h_i} = \sum_{j \leq i} h_j$) qui sont un peu plus coûteux en temps de calcul. Ces deux méthodes sont expliquées plus en détail par Rubner et al. [21] tandis que la troisième sera abordée dans le chapitre 4.

Dès que la signature à utiliser et la métrique associée ont été déterminées, il faut trouver un algorithme qui les emploie afin de localiser une cible dans une image. Nous allons détailler plusieurs de ces techniques dans la section suivante.

2.5 Méthodes de localisation

2.5.1 Méthodes à base de « backprojection »

« Backprojection »

La localisation par « histogram backprojection » ou « rétro-projection d'histogramme » est une technique simple et relativement efficace de localisation de cible grâce à ses caractéristiques de couleur. Cette technique, proposée par Swain et Ballard [23], se base sur la probabilité de chacune des couleurs de l'image analysée d'appartenir à la cible de référence.

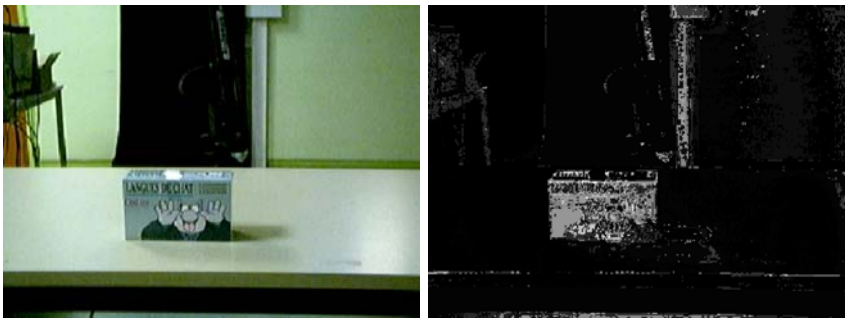
La première étape est de calculer cette probabilité à partir de l'histogramme de la référence (ref) et l'histogramme de l'image (I). Nous définissons pour chaque couleur c :

$$R_c = \min\left(\frac{h_c(ref)}{h_c(I)}, 1\right) \tag{2.13}$$

Ensuite, chaque pixel de l'image (de couleur indexée $h(I_{ij})$) est remplacé par sa probabilité d'appartenir à la cible :

$$I_{ij} = R_{h(I_{ij})} \tag{2.14}$$

Nous obtenons grâce à cela la rétro-projection (figure 2.9). La solution



(a) Image d'origine (b) « Backprojection »

FIG. 2.9 – Exemple de « backprojection »

(x_t, y_t) est trouvée en cherchant le maximum de densité par convolution à partir d'un masque D défini selon la forme et la taille attendues de la cible.

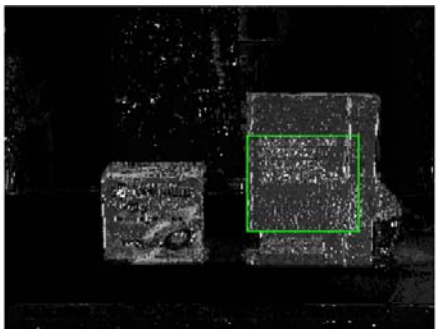
Cette technique a tous les avantages des techniques qui utilisent l'histogramme couleur, à savoir invariance aux orientations, aux translations, aux changements modérés de distance, aux déformations et aux occlusions partielles. De plus, elle demande peu de calculs. Néanmoins, on remarque deux défauts principaux qui rendent son utilisation difficile :

1. sensibilité à l'arrière-fond,
2. sensibilité à une trop forte modification de taille ou d'orientation.

(a) Image avec cible seule

(b) « Backprojection » très prononcée

(c) Image avec objet parasite



(d) Diminution de l'importance de la cible

FIG. 2.10 – Parasitage de la « backprojection »

Ainsi, la « backprojection » est fortement sensible à l'environnement entourant la cible. En effet, si d'autres objets dans la scène ont une des couleurs de la référence, la valeur R_c correspondante diminue. Cela entraîne une diminution de la densité à l'endroit où se trouve la cible. Nous avons donc un objet parasite qui apparaît sur l'image mais qui, en plus, affaiblit la densité

de l'objet-cible. Nous pouvons identifier ce problème sur la figure 2.10 où la boîte de CD's bleue recherchée est bien apparente sur l'image (b). Lorsqu'un livre bleu est placé à côté de la boîte, nous remarquons une forte baisse de la densité de la boîte sur l'image (c). De plus, la localisation (rectangle vert) est faussée puisqu'elle prend le livre comme solution (d). L'algorithme perd donc rapidement la cible quand les couleurs de celle-ci sont trop présentes dans le reste de l'image.

Le second inconvénient vient du fait que le choix d'un masque statique ne permet pas de faire une convolution correcte lorsque la cible dans l'image a une taille ou une orientation fort différente de la référence. Le risque de choisir un objet quelconque en est augmenté. Cependant, des raffinements ont été apportés à la localisation sur la surface rétro-projetée. Ces derniers permettent d'éviter partiellement ces deux problèmes.

« MeanShift »

L'algorithme du « MeanShift » [7] est une extension de la localisation du maximum de densité sur l'image rétro-projetée. Nous ne cherchons plus le maximum de densité global de la surface mais effectuons une recherche itérative à partir d'une position initiale. Pour chaque itération, en suivant le gradient de la distribution de probabilité, nous convergeons vers le pic dominant local.

Pratiquement, le « MeanShift » s'effectue en quatre étapes :

1. choix de la taille de la fenêtre de recherche,
2. définition d'une position initiale de cette fenêtre,
3. calcul du centroïde dans la fenêtre,
4. recentrage sur ce centroïde et, si le déplacement est plus grand qu'une certaine limite, retour en 3, sinon arrêt.

Dans notre cas particulier, la taille de la fenêtre sera définie pour chaque cible de référence et la position initiale correspondra à la position finale de la localisation précédente dans le suivi. Pour calculer le centroïde sur lequel nous nous recentrons, nous utilisons les moments :
Soit M_{00} le moment d'ordre 0,

$$M_{00} = \sum_x \sum_y I(x, y) \tag{2.15}$$

Soient M_{10} et M_{01} les moments de premier ordre pour x et y respectivement,

$$M_{10} = \sum_x \sum_y xI(x,y) \tag{2.16}$$

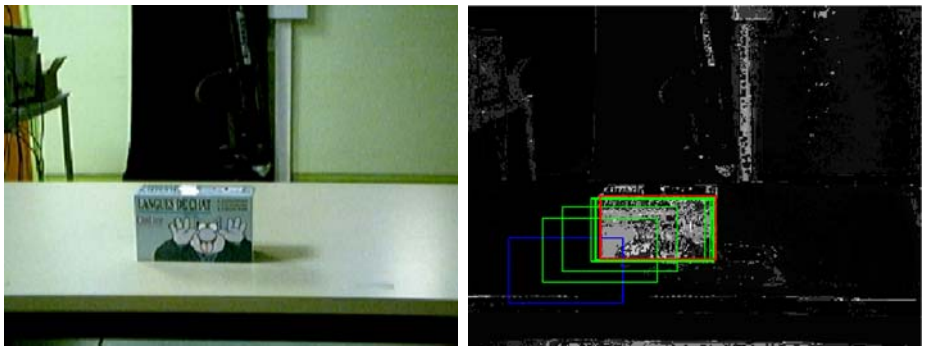
$$M_{01} = \sum_x \sum_y yI(x,y) \tag{2.17}$$

Alors le centroïde de la fenêtre de recherche correspond à

$$x_c = \frac{M_{10}}{M_{00}} \tag{2.18}$$

$$y_c = \frac{M_{01}}{M_{00}} \tag{2.19}$$

Grâce à cette méthode, la fenêtre va se déplacer vers le maximum de densité local (figure 2.11). Sur cette figure, le cadre bleu correspond à la situation d'origine, le vert à des itérations et le rouge à la situation finale.



(a) Image d'origine (b) Itérations

FIG. 2.11 – Exemple d'itérations du « MeanShift »

Sans pour autant résoudre tous les problèmes de la méthode de base, le « MeanShift » évite tout objet parasite qui pourrait être confondu avec la cible puisque aucune recherche globale n'est réalisée. Il garde néanmoins les problèmes liés à la taille de la fenêtre qui reste fixe. Il gère donc toujours assez mal les modifications de taille et d'orientation de l'objet.

Un inconvénient important et difficile à pallier apparaît. Le « MeanShift » est basé sur une recherche locale à partir d'une certaine position initiale. Lorsque cette position est trop éloignée de la cible, le « MeanShift » risque de converger vers un maximum local qui correspond à un objet parasite.

Cela peut se produire lorsque la fenêtre de recherche est placée de façon inadaptée lors de l'analyse de la première image puisque nous n'avons aucune référence précédente. Même en plein suivi, avec la position antérieure de la fenêtre, nous pouvons nous retrouver loin de la cible si elle a bougé trop brusquement dans l'image. Si un objet parasite est choisi par erreur, il sera presque impossible au « MeanShift » de corriger l'erreur initiale.

Certaines solutions ont été proposées dans le cadre du suivi de visages [6], mais elles sont peu générales et ne sauraient être valablement étendues à notre étude. Elle tirent parti des particularités du problème traité pour détecter et éviter le suivi d'objets parasites.

En conclusion, le « MeanShift » améliore la méthode initiale sous certaines conditions. Ainsi, comme expliqué ci-dessus, il est crucial que la cible soit proprement identifiée dès la première image. En outre, des complications surviennent lorsque la cible se déplace trop vite dans les images successives d'un suivi. De plus, certains défauts de la rétro-projection sont toujours présents (dont la faiblesse vis-à-vis des changements importants de taille et d'orientation).

« CamShift »

La procédure du « CamShift » (« Continuously Adaptative Mean Shift »), développée par Bradski [7], perfectionne le fonctionnement du « MeanShift ». Elle permet d'estimer la taille et l'orientation de la cible dans l'image et ainsi d'adapter la taille de la fenêtre de recherche. Pour cela, il faut calculer les moments d'ordre 2 :

$$M_{11} = \sum_x \sum_y xyI(x,y) \tag{2.20}$$

$$M_{20} = \sum_x \sum_y x^2 I(x,y) \tag{2.21}$$

$$M_{02} = \sum_x \sum_y y^2 I(x,y) \tag{2.22}$$

A partir de ces moments, il est possible de calculer l'orientation de la forme, correspondant à l'orientation de l'axe d'inertie principal :

$$\theta = \frac{1}{2} \arctan \left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)} \right) \tag{2.23}$$

Les deux premières valeurs propres (longueur et largeur principales) de la cible peuvent être calculées par :

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \tag{2.24}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \tag{2.25}$$

où

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \tag{2.26}$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \tag{2.27}$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2 \tag{2.28}$$

A partir de w, l et θ, il est possible de redéfinir l'algorithme du « Mean-Shift » afin de modifier la taille de la fenêtre de recherche au fur et à mesure. Cette taille sera toujours augmentée d'une certaine valeur s afin de prendre en compte la possibilité que la cible se soit rapprochée de l'objectif et ait donc grossi. Cette augmentation est définie par Bradski :

$$w = w + s \tag{2.29}$$

$$l = l + s * 1.2 \tag{2.30}$$

où $s = 2\sqrt{M_{00}}$.

La définition du facteur s est dépendante de la dynamique de l'objet à suivre. La décision d'augmenter plus fortement la longueur que la largeur vient du fait que Bradski utilise le « CamShift » afin de suivre des visages. Ces derniers sont en général plus longs que larges.

Conclusion

Les techniques à base de rétro-projection d'histogramme ont un excellent rapport qualité / temps de calcul. Elles sont parfaitement adaptées, comme l'a montré Bradski [7], au suivi de visages. C'est un suivi dont on connaît la dynamique (par exemple, le rapport hauteur / largeur) et pour lequel la cible ne se déplace pas trop brusquement dans l'image. Cependant, les

méthodes de « MeanShift » et « CamShift » sont très sensibles aux erreurs et sont incapables de les corriger. Ces contraintes sont trop fortes dans le cadre des objectifs que nous nous sommes fixés et nous avons donc besoin d'une autre méthode de localisation.

2.5.2 « Colour Histogram Similarity »

Présentation

La méthode de « Colour Histogram Similarity » (CHS par la suite) est une technique de localisation couleur présentée pour la première fois par Swain et Ballard [23]. Le groupe TROP a récemment repris cette approche dans le contexte de l'asservissement visuel robotique [14; 9].

L'algorithme du CHS emploie les histogrammes en tant que signatures d'images. Afin de les comparer, on emploie la distance définie par Swain et Ballard [23] (voir section 2.4.3, équation 2.9).

Le principe est simple. Il faut d'abord définir une cible d'une taille quelconque sur une image (zone rectangulaire) ; l'histogramme M de cette zone de l'image est calculé. Il convient d'analyser ensuite l'image IMG sur laquelle on désire localiser l'objet-cible. Pour cela, une fenêtre de recherche FR, de la taille de la cible, est utilisée pour parcourir IMG pixel par pixel. Ainsi, pour chaque pixel P, l'histogramme $H(P)$ de la fenêtre de recherche (le coin supérieur-gauche de FR se trouvant à la position P) est calculé. L'étape finale consiste à appliquer l'intersection d'histogrammes sur $H(P)$ et M pour en déduire une mesure de similarité $C(P)$. L'ensemble des $C(P)$ forme la surface de similarité (de taille $(L_1 - l_1 + 1) \times (L_2 - l_2 + 1)$, où l'image IMG est de taille $L_1 \times L_2$ et FR est de taille $l_1 \times l_2$). L'algorithme considère alors que la position (coin supérieur-gauche) de la cible se trouve là où la surface est maximale c'est-à-dire là où la similarité est maximale.

La figure 2.12 illustre le fonctionnement du CHS. Les flèches vertes indiquent un déplacement de la fenêtre de recherche. Après chaque mouvement, les deux histogrammes sont comparés pour construire pixel par pixel une surface de similarité telle que celle de la figure 2.13. La signification des couleurs de cette surface va être expliquée ci-après.

D'après la définition de Swain et Ballard [23], on peut déduire que les $C(P) \in [0, 1]$. Il est donc facile de représenter graphiquement la surface de similarité. Pour cela, nous avons utilisé une palette qui est en fait une

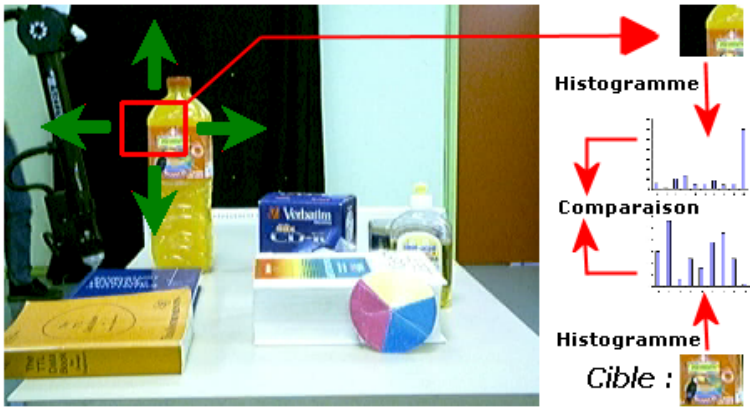


FIG. 2.12 – Principe de l'algorithme du CHS

gradation partant du bleu foncé, passant par des tons de bleu, cyan, vert, jaune et rouge, et qui se termine par du rouge foncé. Ainsi, un pixel bleu a une faible probabilité d'être la position du coin supérieur gauche de la cible dans l'image, contrairement à un pixel rouge. La figure 2.13 montre clairement les avantages d'un tel procédé, car on repère directement les pics, ce qui est très utile lors de la mise au point d'un algorithme.

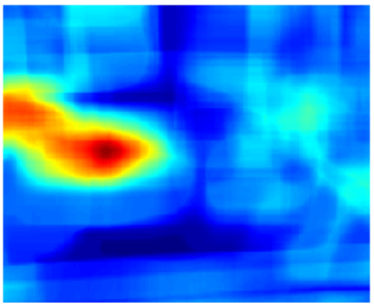


FIG. 2.13 – Exemple de surface de similarité

Résultats et limitations

L'algorithme du CHS fournit d'assez bons résultats [14; 9]. Il se montre d'ailleurs plus performant que la méthode de la « backprojection ».

Il y a cependant quelques limitations inhérentes aux techniques utilisées :

– l'histogramme de la cible doit être assez discriminant,

- les couleurs de l'objet dans les images analysées doivent rester proches de celles du modèle recherché,
- la taille de l'objet et du modèle sont supposées être identiques.

La première limitation concerne l'objet à localiser. Il est nécessaire que les couleurs de ce dernier soient suffisamment contrastées par rapport au reste de la scène. Effectivement, comme nous ne basons l'algorithme que sur une signature couleur, si la cible ne se distingue pas clairement des autres objets présents, des résultats peu satisfaisants seront obtenus. L'intégration d'autres informations dans la signature (espace, texture, ...) pourrait corriger cet inconvénient, mais la contrainte de temps réel limite les possibilités envisageables.

La seconde restriction est quant à elle très contraignante et ne saurait être totalement vérifiée. Ainsi, les changements d'illumination, même minimes, ont des conséquences fâcheuses et les « bins » de l'histogramme du modèle ne correspondront plus à ceux de l'image à analyser.

Le dernier aspect concerne les changements de taille de la cible dans l'image par rapport au modèle. En pratique, de légères variations ne posent pas trop de soucis : l'objet recherché est toujours localisé malgré une perte de précision.

Il nous reste finalement à remarquer que les inconvénients du CHS sont également présents dans l'algorithme de la « backprojection ».

Performances

Grâce aux techniques employées (intersection d'histogrammes, balayage de l'image), le temps de calcul est peu élevé. Néanmoins, certaines optimisations sont requises si on désire atteindre des calculs en temps réel (25 images par seconde).

Une première optimisation a été réalisée au niveau du déplacement de la fenêtre de recherche. Lors de ce mouvement, au lieu de recalculer l'histogramme complet de la zone, il suffit de ne plus tenir compte des pixels qui n'en font plus partie et d'ajouter l'apport des autres. La figure 2.14 illustre cette optimisation. En effet, lors d'un déplacement de gauche à droite sur l'image, il suffit de supprimer la contribution à l'histogramme des pixels de la première colonne et d'ajouter celle de la colonne suivant la dernière de la fenêtre. Un principe similaire est utilisé lors de déplacement de droite à gauche

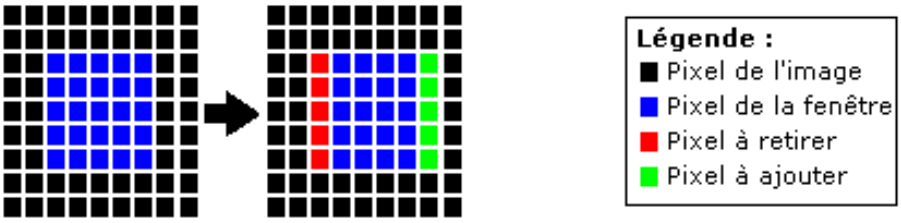


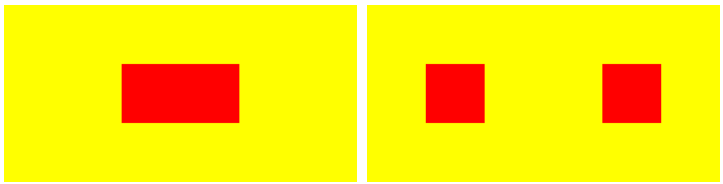
FIG. 2.14 – Déplacement de la fenêtre de recherche

et de haut en bas. Ainsi, la fenêtre de recherche serpente sur l'ensemble de l'image.

Une autre technique pour gagner du temps de calcul consiste à employer les instructions SSE2 (spécifiques aux processeurs Intel Pentium IV). Ce jeu d'instructions permet d'effectuer aisément des calculs vectoriels, ce qui facilite le traitement simultané de plusieurs pixels. Grâce à ces optimisations, l'algorithme détecte l'objet recherché en 10-15 ms[7] (cela dépend également de la taille de la cible), ce qui laisse une marge pour remédier aux problèmes cités ci-dessus.

2.5.3 Corrélogramme couleur

L'utilisation de l'histogramme couleur comme signature d'objet est en général suffisante pour le différencier d'un autre. Nous pouvons cependant rappeler qu'il suffit que deux objets aient les mêmes proportions de couleurs pour qu'ils aient un histogramme identique (figure 2.15).



(a) Image n°1 (b) Image n°2

FIG. 2.15 – Exemple d'images différentes ayant le même histogramme

[7] Temps calculé sur un ordinateur doté d'un processeur Pentium IV Xeon 2.4Ghz (bi-processeur) avec 512Mo de RAM, tournant sous Windows XP.

Nous pouvons voir sur cet exemple que, même si les proportions de couleurs sont identiques, la disposition des pixels de couleur est différente. Nous pourrions donc essayer d'incorporer une information sur la distribution spatiale locale (information de texture) des couleurs dans l'image. Plusieurs approches ont été essayées à cette fin, dont celle de Chang et Krumm [10] avec l'histogramme de co-occurrences et celle de Huang [15] avec le corrélogramme de couleur. Chacune de ces approches apporte une information de texture à l'histogramme de base. Ce nouvel histogramme, dans le cas de Huang [15], s'appelle le corrélogramme.

Pour chaque couleur c_i, le corrélogramme représente la probabilité de trouver la couleur c_j à une distance k en pixels. La distance entre deux pixels $p_1 = (x_1, y_1)$ et $p_2 = (x_2, y_2)$ est définie par la norme L_∞ c'est à dire $|p_1 - p_2| = \max(|x_1 - x_2|, |y_1 - y_2|)$ (ainsi les huit pixels entourant tout pixel p auront une distance de 1). A partir de cette distance, nous pouvons définir le corrélogramme comme :

$$\gamma^{(k)}_{c_i,c_j}(I) = Pr[p_2 \in I_{c_j}, |p_1 - p_2| = k | p_1 \in I_{c_i}] \tag{2.31}$$

où I_{c_j} représente l'ensemble des pixels de couleur c_j dans l'image I.

Pour calculer cette valeur, il suffit de, premièrement, calculer le nombre de pixels de couleur c_i à une distance k de pixels de couleur c_j :

$$\Gamma^{(k)}_{c_i,c_j}(I) = |\{p_1 \in I_{c_i}, p_2 \in I_{c_j}, |p_1 - p_2| = k\}| \tag{2.32}$$

Ensuite déduire γ :

$$\gamma^{(k)}_{c_i,c_j}(I) = \frac{\Gamma^{(k)}_{c_i,c_j}(I)}{h_{c_i}(I) 8k} \tag{2.33}$$

Le dénominateur correspond au nombre maximal de pixels voisins à une distance k pour un nombre quelconque de pixels de couleur c_i. Le facteur $8k$ équivaut au nombre de pixels se trouvant à une distance k dans la norme L_∞.

γ est calculé pour chaque couple de couleurs (indexées) et pour chaque k compris entre 1 et un paramètre d fixé. Si la quantification est de n « bins », la taille du corrélogramme sera $O(n^2 d)$. Cette taille, qui peut devenir assez grosse si le « binning » est fin, peut être réduite en ne considérant que les

couples de couleur identique. Nous obtenons de cette manière l'autocorrélogramme :

$$\alpha_c^{(k)}(I) = \gamma_{c,c}^{(k)}(I) \tag{2.34}$$

Ainsi, sa taille n'est plus que $O(nd)$, ce qui est beaucoup plus raisonnable. L'utilisation de l'autocorrélogramme sur les images de la figure 2.15 permet de remarquer que les deux images ont bien deux corrélogrammes différents (figure 2.16). On peut y remarquer une nette différence dans les courbes qui représentent la couleur rouge. Les probabilités pour les distances 1, 2 et 3 sont plus importantes pour l'image n°1 puisque l'ensemble des pixels rouges sont regroupés. Les probabilités pour les distances 5, 6 et 7 sont, par contre, plus grandes pour l'image n°2 puisque nous avons une dispersion en deux groupes de pixels rouges.

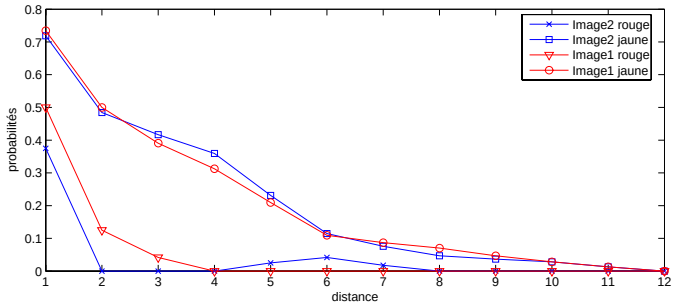


FIG. 2.16 – Valeurs de l'autocorrélogramme pour les images de la figure 2.15

L'intersection de deux corrélogrammes peut être réalisée exactement de la même manière que pour deux histogrammes. Par exemple, en utilisant la norme L_1 :

$$|I - I'| = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{d} |\gamma_{c_i,c_j}^{(k)}(I) - \gamma_{c_i,c_j}^{(k)}(I')| \tag{2.35}$$

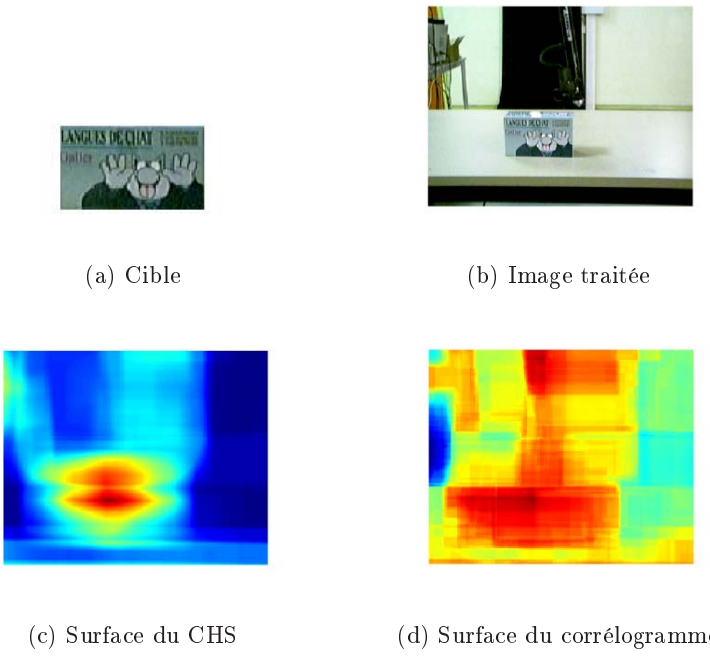
Pour l'autocorrélogramme :

$$|I - I'| = \sum_{i=1}^{n}\sum_{k=1}^{d} |\alpha_{c_i}^{(k)}(I) - \alpha_{c_i}^{(k)}(I')| \tag{2.36}$$

Nous avons donc une méthode pour extraire les informations de distribution spatiale locale des couleurs dans l'image et un moyen pour les comparer. Le temps de calcul nécessaire à toutes ces opérations est relativement lourd

en comparaison au CHS. En effet, Huang a créé cette technique afin d'effectuer la recherche d'une image de référence dans une base d'images prédéfinies. Ce problème a des contraintes différentes de celles du suivi en temps réel et n'a pas besoin de calculer les corrélogrammes pour chaque comparaison. Les images de la base de données sont déjà transformées en corrélogramme et chaque requête ne demande plus qu'un seul calcul pour la référence et une intersection pour chaque image dans la base.

Dans notre cas, nous avons autant d'intersections que de générations de corrélogrammes. Si la comparaison d'autocorrélogrammes avec la norme L_1 est assez rapide, les calculs nécessaires pour constituer le corrélogramme sont assez lourds. La technique la plus directe serait de considérer chaque pixel de l'image de couleur c_i et, pour chaque k de 1 à d, compter le nombre de pixels de couleur c_j qui résolvent $|p_1 - p_2| = k$. Si l'image est de taille $n \times m$, cela prendrait un temps en $O(nmd^2)$. Huang propose une version plus élaborée en $O(nmd)$. Malgré cela, avec nos contraintes de temps réel, il est difficile de mettre le corrélogramme en oeuvre. Avec un paramètre $k = 3$, le temps de calcul nécessaire au traitement d'une image de 320x240 avec une cible de 81x46 est dix fois supérieur à celui du CHS.



(a) Cible (b) Image traitée

(c) Surface du CHS (d) Surface du corrélogramme

FIG. 2.17 – Comparaison de résultats entre le CHS et le corrélogramme

De plus, les tests réalisés sur des images concrètes n'ont montré aucun avantage en faveur du corrélogramme (figure 2.17). Nous n'avons malheureusement pas eu l'occasion de déterminer les causes de ce résultat insatisfaisant au cours de ce travail. Il est possible que dans la plupart des cas, la distribution des couleurs soit suffisante pour définir un objet. En effet, si un objet est composé d'un nombre suffisant de couleurs distinctes, l'histogramme représente correctement notre cible. Dans la suite de cette étude, nous ne considérerons donc plus que le CHS.

2.6 Conclusion

Tout au long de ce chapitre, nous avons donc présenté les concepts essentiels à la compréhension du problème de la localisation couleur.

Pour cela, nous avons d'abord introduit des notions de base liées à la couleur, telles que la luminosité, la teinte, la saturation, ...

Ensuite, nous avons montré que la comparaison d'images en couleur, de part leur taille en mémoire, nécessitait un traitement préalable : la détermination de leur signature. Celle-ci doit être simple et peu gourmande en ressources tout en préservant le maximum d'informations sur l'image. Cela nous a conduit à définir le concept d'histogramme couleur, une signature très utilisée dans le domaine du traitement d'images en couleurs. Après cela, nous avons présenté des fonctions de mesure de la distance entre deux images, utilisées pour en calculer le degré de similitude.

Enfin, différents algorithmes de localisation couleur ont été expliqués en détail. Parmi ceux-ci, le CHS, sur lequel le TROP base ses recherches, fut le point de départ de la plupart de nos travaux lors de notre stage en leurs bureaux.

Le CHS a démontré qu'il avait du potentiel et ses performances laissent place à des possibilités d'amélioration tout en restant dans les limites des applications en temps réel. Nous avons vu qu'il fournissait les meilleurs résultats tout en respectant les contraintes que nous nous étions fixées. Nous avons expliqué ses faiblesses, liées notamment à l'utilisation de l'histogramme comme signature. Nous les rappelons brièvement ci-après :

– l'histogramme de la cible doit être assez discriminant,

- les couleurs de l'objet dans les images analysées doivent rester proches de celles du modèle recherché,
- les tailles de l'objet et du modèle sont supposées être identiques.

Nous allons nous attarder sur ces problèmes dans les chapitres suivants. Nous y détaillerons des techniques permettant de pallier aux défauts du CHS et nous proposerons à chaque fois des résultats de tests. Ainsi, le second problème, surtout dû à des changements d'illumination, sera abordé dans notre étude des espaces couleur (chapitre 3) et il sera étudié en profondeur dans la partie dédiée à la robustesse à l'illumination (chapitre 4). Dans le chapitre 5, nous tenterons de résoudre le troisième problème. Effectivement, celui-ci est très important puisque l'on envisage la localisation couleur dans un but de déplacement robotique (l'objet grandissant au fur et à mesure que le robot s'approche). La dernière partie, les indicateurs, abordera la question de l'évaluation des résultats. Il s'agira en fait de savoir à quel point la position détectée est correcte.

Par contre, nous n'étudierons pas le premier problème qui est inhérent à l'utilisation des histogrammes. Une solution serait d'incorporer d'autres données dans la signature, mais le coût engendré en temps de calcul semble trop important par rapport au gain potentiel.

Analyse de la problématique

Chapitre 3

Espaces couleur

3.1 Introduction

Ce chapitre est consacré à l'étude de différents espaces de représentation de la couleur. Un minimum de connaissances au sujet de ces derniers est absolument nécessaire si l'on désire effectuer du traitement d'images. Effectivement, les résultats d'un algorithme peuvent être radicalement différents selon que l'on utilise l'un ou l'autre espace.

L'algorithme du CHS se base sur l'utilisation d'histogrammes couleur. Or, la création d'un histogramme suppose une partition de l'espace des couleurs. Dès lors, on peut se demander quelle est l'influence de la partition utilisée sur les performances du CHS, puisque celui-ci utilise exclusivement l'information couleur pour localiser la cible.

Pour cela, nous allons réaliser dans cette section un aperçu des différents modes de codage de la couleur. Il en existe un nombre assez conséquent, ceux-ci ayant été introduits au fil du temps par des scientifiques en fonction des besoins (par exemple, la famille du YC_bC_r est utilisée sur nos téléviseurs couleur, car elle permet une compatibilité arrière avec les téléviseurs noir et blanc).

Il existe des espaces qui nécessitent la prise en compte de certains paramètres tels que l'illuminant et la correction gamma afin d'en assurer un usage optimal. Beaucoup de travaux se focalisent sur la qualité de reproduction des couleurs pour l'oeil humain. Au contraire, notre objectif est d'améliorer la localisation d'un objet dans une scène. C'est pourquoi nous ne ferons pas ci-après une analyse exhaustive, mais nous nous contenterons de faire une

brève description des espaces les plus connus — et donc, les plus utilisés —, accompagnée des formules de conversion depuis l'espace RGB.

Nous réaliserons notre étude en suivant la répartition proposée par Tremeau et al. [24]. Selon leur point de vue, il existe quatre types de systèmes de représentation de la couleur :

- les systèmes de primaires,
- les systèmes luminance-chrominance,
- les systèmes perceptuels,
- les systèmes d'axes indépendants.

Ces quatre familles ne constituent pas une partition pour les espaces. En effet, chacune d'entre elles dispose de caractéristiques intéressantes. Dès lors, durant l'élaboration d'un nouveau mode de représentation de la couleur, les scientifiques ont souvent tenté d'en combiner les propriétés. Ainsi, il n'est pas rare qu'un espace couleur puisse appartenir à plusieurs familles. Le plus grand point commun entre eux est certainement le fait que chaque couleur est définie par le biais de trois composantes.

Parmi ceux de la catégorie des systèmes de primaires, nous retrouvons RGB, CMY et XYZ. Il s'agit de systèmes basés surtout sur le principe de la trivariance visuelle qui a déjà été décrit plus tôt dans ce document (section 2.2.1).

Dans la catégorie des systèmes luminance-chrominance, on peut discerner plusieurs sous-groupes qui se différencient par la manière de calculer les coordonnées de luminance et de chrominance :

- Les systèmes de type YC_bC_r correspondent à des standards de télévision. Nous étudierons le système générique YC_bC_r et deux de ses applications, le YIQ et le YUV.
- Les systèmes antagonistes tels que le AC_1C_2 se basent sur la théorie des couleurs opposées de Young et Herin [24]. Selon celle-ci, l'information couleur captée par nos sens est transmise au cerveau par le biais de trois signaux : les oppositions noir-blanc, vert-rouge et bleu-jaune. La plupart des systèmes de cette famille sont peu utilisés en imagerie couleur, car ils sont soit trop simplifiés, soit expérimentaux.

Parmi les systèmes perceptuels, nous étudierons notamment $L^*a^*b^*$. Leur but est de rester uniforme au sens de la perception visuelle humaine. En

d'autres mots, l'écart entre deux couleurs proches dans l'espace est proportionnel à la différence avec laquelle nos yeux les perçoivent. Ces systèmes comprennent également les systèmes géométriques, tels que le HSV, qui représentent les composantes d'une couleur de façon géométrique.

Enfin, les systèmes d'axes indépendants sont des espaces dans lesquels la corrélation entre les différentes composantes couleur est nulle. En effet, suivant la distribution des couleurs considérée (en d'autres termes, suivant l'image étudiée et l'espace utilisé), il peut y avoir une corrélation plus ou moins importante entre les composantes couleur. Si cette dernière est forte, traiter chaque composante indépendamment des autres revient à perdre l'information commune aux trois. C'est pourquoi les systèmes d'axes indépendants ont été introduits, nous décrirons brièvement les espaces $X_1X_2X_3$ et $I_1I_2I_3$.

Il faut préciser que les formules trouvées dans la littérature sont rarement directement utilisables dans un programme. Par exemple, certains espaces n'ont pas été créés dans un but informatique et la plage des valeurs disponibles pour les composantes d'une couleur risque ainsi de ne pas correspondre à l'utilisation que nous en faisons[1] (stockage sur un octet par composante, valeur entre 0 et 255). Afin de remédier à ce problème, nous avons dû adapter les formules de conversion (mise à échelle, décalage, ...). Bien évidemment, cela entraîne une dégradation de l'espace (amoindrissement de la gamme des couleurs représentables). Néanmoins, puisque le temps nous manquait pour étudier cela en détail, nous avons décidé de ne pas tenir compte de cette détérioration.

Dès le départ, c'est-à-dire lors de l'acquisition de l'image, nous perdons de l'information par rapport à l'oeil humain. En effet, il a été démontré que, pour en atteindre le niveau de sensibilité, il faudrait coder chaque composante sur 10 bits [25] (soit 2 bits de plus qu'avec du matériel standard). Donc, quel que soit l'espace utilisé, toutes les couleurs que nous pouvons percevoir ne sont pas représentables. Nous verrons cependant que ce phénomène est aggravé dans certains espaces.

De plus, un second problème tout aussi considérable intervient : la quantification. Comme expliqué plus tôt dans ce document, avant d'appliquer

[1] Cet inconvénient est apparu lors de tests dans le cadre de notre application, mais il est cependant valable pour toute utilisation informatique de ces espaces puisqu'ils n'ont pas été conçus dans une telle optique.

l'algorithme du CHS sur une image, nous en réduisons le nombre de couleurs par quantification afin de diminuer la taille de l'histogramme, et donc réduire les temps de calculs. Mais la question qui se pose ici est de savoir quel poids attribuer à chaque composante lors de cette opération. Nous reviendrons sur ce problème lors de l'étude comparative qui termine cette partie.

Nous conclurons notre survol des espaces couleur par quelques comparaisons. En effet, le temps étant crucial dans l'optique que nous suivons, nous ne devons pas négliger le temps requis par le passage du *RGB* dans l'espace de destination. Nous effectuerons également des tests de l'algorithme du CHS sur chaque mode de représentation afin d'évaluer les avantages potentiels. Tous ces tests nous permettront de dégager des espaces couleur qui amélioreront éventuellement les performances du CHS.

3.2 Systèmes de primaires

3.2.1 *RGB*

L'espace *RGB* est le plus couramment utilisé en informatique. Il l'est notamment pour les moniteurs couleur et la plupart des caméras vidéo. Il est donc logiquement utilisé par les applications, car aucune transformation n'est requise pour afficher le résultat à l'écran. Généralement, des périphériques ou logiciels utilisent d'autres espaces de couleur lorsque le marché visé est plus professionnel. Néanmoins, étant donné les hypothèses de travail que nous nous sommes fixées, le matériel que nous utiliserons sera de type standard et les données seront donc au format *RGB*. Par conséquent, lors des expériences, si nous désirons travailler avec un autre espace de couleur, nous devrons effectuer une conversion préalable.

Dans l'espace *RGB* (R pour « red », G pour « green » et B pour « blue »), chaque couleur est décrite en termes de quantité de rouge, de vert et de bleu. La plupart des couleurs peuvent être obtenues grâce à un mélange de ces trois couleurs. Ces dernières sont donc les primaires de cet espace. Il s'agit d'un modèle additif : les couleurs s'additionnent afin d'en former de nouvelles.

Il faut préciser qu'il n'y a pas un seul espace *RGB*, mais bien plusieurs. Effectivement, l'espace dépend des primaires R, G et B et du blanc de référence utilisé. Pour travailler de manière précise, il faudrait déterminer les primaires utilisées par la caméra et l'illuminant utilisé lors de l'étape d'ac-

quisition[2]. Ensuite, il faudrait faire correspondre aux valeurs RGB obtenues celles de l'espace dans lequel on compte travailler. Néanmoins, comme cela a déjà été précisé dans ce document, vu nos hypothèses de travail, nous ne pouvons pas tenir compte de ces paramètres pour des raisons de matériel et de temps de calcul.

Deux problèmes surviennent lors de l'utilisation de l'espace RGB pour du traitement d'images :

- Il n'est pas linéaire avec la perception visuelle. Autrement dit, la perception de la différence entre deux couleurs varie en grandeur dans l'espace. Ainsi, une faible différence d'éclairage entre deux images n'implique pas nécessairement que les couleurs de deux pixels de même position dans chacune d'elles soient voisines dans l'espace RGB.
- Il y est impossible de représenter toutes les couleurs qui nous sont visuellement perceptibles. Effectivement, à chaque primaire correspond une longueur d'onde. Ainsi, dans l'espace CIE RGB 1931, il s'agit de 700.0 nm pour le rouge, 546.1 nm pour le vert et 435.8 nm pour le bleu. Lorsqu'on effectue l'expérience d'égalisation (décrite dans la section 2.2.1) pour ces trois primaires, on se rend compte que certaines couleurs que nous percevons auraient dans cet espace une ou plusieurs composantes à valeur négative. Les coefficients trouvés lors de cette expérience ne peuvent donc pas être employés tels quels pour décrire une couleur dans un programme informatique.

Ce second inconvénient est plus facile à comprendre en se référant à la figure 3.1. Nous pouvons y observer les fonctions colorimétriques du système CIE RGB 1931 pour les couleurs du spectre visible (longueur d'onde comprise entre 400 et 700 nm). Ces fonctions représentent en fait les coefficients obtenus lors de l'expérience d'égalisation. Pour rappel, pour les trois primaires R_C, G_C et B_C et une couleur C_λ, cette expérience permettait de calculer les coefficients r, g et b tels que :

$$C_\lambda = r(\lambda)\,[R_C] + g(\lambda)\,[G_C] + b(\lambda)\,[B_C]$$

Chaque primaire est obtenue en annulant les coefficients des deux autres primaires. Par exemple, pour la primaire $[G_C]$, $r(\lambda)$ et $b(\lambda)$ valent 0 à la longueur d'onde $\lambda = 546,1$ nm. Précisons également que les courbes obtenues

[2] L'illuminant et le blanc de référence sont deux variables liées, la seconde étant déterminée par la première (voir sections 2.2.6 et 2.2.7).

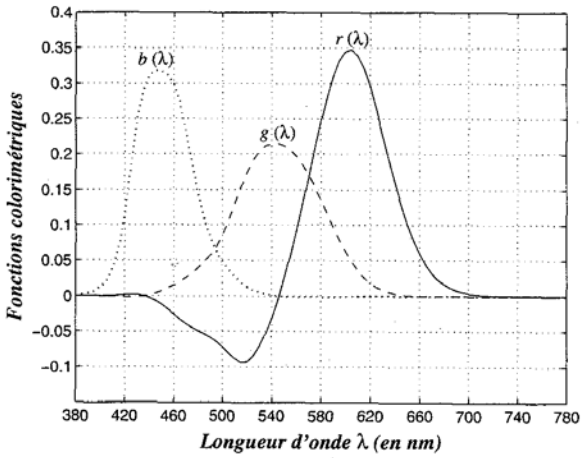


FIG. 3.1 – Fonctions colorimétriques du système RGB (source : [24])

sont normalisées sur la figure 3.1 (les surfaces sous chacune des courbes sont égales). Les coefficients de pondération valent respectivement 1.0000, 4.5907 et 0.0601.

Enfin, on peut aussi remarquer que l'espace RGB est fortement corrélé. On peut dire qu'un espace est corrélé lorsqu'une information commune est contenue partiellement dans chaque composante. Ainsi, dans le cas du RGB, l'information de luminance est répartie sur les trois composantes. La corrélation du RGB n'est pas à son avantage puisqu'un traitement séparé des composantes ne peut être employé sans perte d'information.

3.2.2 CMY

L'espace CMY (C pour « cyan », M pour « magenta » et Y pour « yellow ») suit une philosophie opposée au RGB. Effectivement, contrairement à ce dernier, le CMY est de type soustractif. Il se base sur les propriétés d'absorption de la lumière dont fait preuve l'encre sur du papier. Ainsi, lorsque de la lumière blanche atteint l'encre, certaines longueurs d'onde visibles sont absorbées tandis que d'autres sont réfléchies. On peut donc assimiler cela à un système de filtres.

Il existe également un espace dérivé, le $CMYK$ (K pour « black »), qui a été introduit dans le but d'augmenter la gamme de couleurs représentables. La raison de cet apport est que la combinaison de purs cyan, magenta et

jaune devant produire une lumière donnant lieu à du noir produisait plutôt du brun. Il a donc fallu ajouter le noir comme primaire.

Cet espace souffre des mêmes défauts que le RGB et de plus, il n'est pas très intuitif. Il est difficile de se faire mentalement une idée d'une couleur en en soustrayant d'autres à des primaires. Enfin, d'après Ford et Roberts [13], une conversion correcte du RGB vers le CMY est assez complexe[3]. Les paramètres qui entrent en compte concernent notamment l'impression, ce qui ne correspond pas à l'usage que nous faisons du CMY. Une transformation simplifiée proposées par Ford et Roberts [13] est la suivante :

$$\begin{pmatrix} Cyan \\ Magenta \\ Yellow \end{pmatrix} = \begin{pmatrix} 255 - Red \\ 255 - Green \\ 255 - Blue \end{pmatrix} \tag{3.1}$$

D'autres auteurs, dont notamment Tremeau et al. [24], préconisent une autre transformation, tout aussi simpliste :

$$\begin{pmatrix} Cyan \\ Magenta \\ Yellow \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} Red \\ Green \\ Blue \end{pmatrix} \tag{3.2}$$

3.2.3 XYZ

Le système XYZ (aussi appelé CIE XYZ 1931) est le système colorimétrique de référence défini par la CIE. Il a été conçu de façon à ce que toute couleur visible puisse être définie en utilisant uniquement des valeurs positives, il s'agit donc de pallier un des défauts de l'espace RGB.

La conversion du RGB vers le XYZ nécessite la prise en compte de nombreux paramètres. Parmi ceux-ci, il y a surtout le blanc de référence. Ce dernier correspond généralement à un illuminant normalisé (ceux-ci ont été définis dans la section 2.2.6). L'illuminant D_{65} est sans doute le plus utilisé, il correspond à la moyenne arithmétique des lumières durant la journée. C'est ce dernier qui sera considéré lorsqu'un illuminant devra être pris en compte dans des calculs.

[3] Cet espace étant principalement destiné à l'impression, une conversion correcte devrait tenir compte de paramètres tels que le type d'encre et le type de papier utilisés ou encore le type de périphérique d'impression. C'est d'ailleurs pour cela que les logiciels professionnels utilisent, lors de l'impression, des profils ICC, définis par les fabricants, qui spécifient les valeurs de ces paramètres.

L'espace XYZ, tout en corrigeant un des problèmes principaux du RGB, dissocie les informations de luminance (composante Y) et de chrominance (composantes X et Z). Cependant, son défaut réside dans le fait que, pour une utilisation optimale, il faut connaître les conditions d'acquisition, ce qui n'est pas notre cas. En outre, comme le RGB, il est non linéaire avec la perception visuelle.

La conversion RGB vers XYZ est une opération qui s'effectue via une multiplication matricielle de la forme suivante :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.3}$$

Les coefficients (X_r, Y_r, Z_r), (X_g, Y_g, Z_g) et (X_b, Y_b, Z_b) sont respectivement les coordonnées X, Y et Z des primaires rouge, vert et bleu considérées. Comme précisé ci-dessus, ils dépendent également du blanc de référence.

Dans la suite, nous considérerons une conversion depuis l'espace nommé *Rec. 709 RGB* (correspondant aux écrans CRT contemporains, c'est-à-dire aux écrans standards d'ordinateurs) vers l'espace CIE XYZ en utilisant l'illuminant D_{65} comme blanc de référence. Pour cela, nous utiliserons la formule donnée par Poynton [20] :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.4}$$

Parmi les nombreuses sources consultées, les formules étaient souvent différentes. Il semble que peu de personnes effectuent cette conversion en toute connaissance de cause : elles fournissent leur matrice de conversion sans préciser les paramètres pris en compte. Dans notre cas, nous avons choisi la formule ci-dessus pour les raisons suivantes :

- L'environnement dans lequel la caméra et l'algorithme du CHS sont destinés à être utilisés n'étant pas défini, nous avons donc choisi l'illuminant standard le plus commun.
- La caméra procède automatiquement à des corrections au niveau de la lumière, cet illuminant est donc susceptible de changer sans que nous n'ayons de contrôle. Cela peut d'ailleurs poser problème lors de la localisation de la cible (voir chapitre 4). Ce choix est donc totalement arbitraire.

- Calculer les coefficients de cette matrice avec précision nécessite de connaître certaines caractéristiques techniques du matériel utilisé. Or, vu nos hypothèses de travail (matériel bon marché), nous n'avons pas accès à ces informations. L'alternative consiste à mesurer soi-même ces caractéristiques, mais cela requiert un matériel relativement coûteux.
- La caméra étant destinée au grand public, on peut supposer qu'elle soit calibrée de façon à ce que les images affichées à l'écran soient les plus fidèles possibles par rapport à la capture faite. On peut donc supposer que les coefficients corrects sont proches de ceux d'un écran CRT.

3.3 Systèmes luminance-chrominance

3.3.1 Systèmes de type YC_bC_r

YC_bC_r est le standard international de codage pour la télévision. Il a été développé à l'origine pour maintenir une compatibilité avec les écrans noir et blanc lors de l'apparition de la couleur dans les postes de télévision. La coordonnée Y contient des informations de luminance tandis que les deux autres décrivent la chrominance. Ainsi, seule la première est utilisée dans le cas d'un écran monochrome.

La séparation des informations de chrominance et de luminance permet de diminuer la corrélation entre composantes par rapport à l'espace RGB. Cependant, ces espaces sont principalement utilisés dans le cadre des images TV et ils sont peu intuitifs, à moins d'être ingénieur en télévision.

La conversion se fait par une transformation linéaire dont les coefficients déterminent l'espace de destination. En effet, les modèles YIQ et YUV sont assez semblables, ils ne diffèrent que par les valeurs de ces coefficients. Chaque modèle est utilisé pour un standard de télévision différent (YUV pour le PAL, YIQ pour le NTSC, ...). L'espace YC_bC_r ITU.BT-601 est un standard international de codage digital d'images TV.

Voici la formule générique utilisée pour la conversion vers ces espaces :

$$\begin{aligned} Y &= 0.299 * R + 0.587 * V + 0.114 * B \\ C_b &= a_1(R - Y) + b_1(B - Y) \\ C_r &= a_2(R - Y) + b_2 * (B - Y) \end{aligned} \tag{3.5}$$

Comme dit précédemment, les coefficients a_1, b_1, a_2 et b_2 dépendent de l'espace considéré. La composante Y est donc identique dans tous les espaces de cette famille et également à la composante Y du système CIE XYZ. Il faut cependant préciser qu'une correction gamma, différente pour chaque système, est appliquée sur les valeurs RGB. Nous n'en tiendrons néanmoins pas compte dans ce qui suit.

Voici les formules qui seront utilisées pour passer respectivement dans l'espace YC_bC_r ITU.BT-601 (donnée par Ford et Roberts [13]), YUV et YIQ :

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & 0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.6}$$

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.7}$$

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.8}$$

Il faut préciser qu'il est inutile d'effectuer de tests sur l'espace YUV, car, suite à l'adaptation nécessaire afin de récupérer des valeurs comprises entre 0 et 255, la formule de conversion est identique à celle utilisée pour l'espace YC_bC_r ITU.BT-601. Enfin, YIQ est très semblable au YUV, les plans UV et IQ diffèrent d'une rotation de 33°.

Enfin, aux conversions expliquées ci-dessus, il faut ajouter une opération de correction gamma que nous négligerons lors de nos tests. En effet, cette dernière dépend de paramètres que nous ne maîtrisons pas. En outre, nous utilisons ces espaces uniquement à des fins de calcul et non à des fins d'affichage.

3.3.2 Systèmes antagonistes

Comme cela a été dit, les systèmes antagonistes se basent sur la théorie des couleurs opposées de Young et Hering. Selon cette dernière, notre cerveau reçoit les informations relatives à la couleur sous forme de trois signaux qui correspondent à des oppositions noir-blanc, rouge-vert et bleu-jaune.

Parmi les espaces couleur appartenant à cette famille, AC_1C_2 est le plus utilisé en imagerie numérique. Il est en fait un des espaces les plus proches de la perception humaine en ce qui concerne la couleur. A est la composante achromatique du modèle. Les deux autres correspondent respectivement aux oppositions rouge-vert et bleu-jaune.

La formule de conversion proposée par Tremeau et al. [24] se déroule en trois étapes :

- conversion RGB vers XYZ,
- conversion XYZ vers LMS,
- conversion LMS vers AC_1C_2.

La première étape ayant été développée plus tôt dans ce document, voici le détail des deux autres :

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.15514 & 0.54312 & -0.03286 \\ -0.15514 & 0.45684 & 0.03286 \\ 0 & 0 & 0.00801 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{3.9}$$

$$\begin{aligned} A &= a \times (\alpha \times \log(L) + \beta \times \log(M) + \gamma \times \log(S)) \\ C_1 &= u_1 (\log(L) - \log(M)) \\ C_2 &= u_2 (\log(L) - \log(S)) \end{aligned} \tag{3.10}$$

Les différents coefficients varient suivant le modèle perceptuel considéré. Nous utiliserons les valeurs du modèle de Faugeras citées par Tremeau et al. [24] : $a = 22.6$, $\alpha = 0.612$, $\beta = 0.369$, $\gamma = 0.019$, $u_1 = 64$, $u_2 = 10$. Cela donne donc la formule suivante :

$$\begin{aligned} A &= 22.6 \times (0.612 \times \log(L) + 0.369 \times \log(M) + 0.019 \times \log(S)) \\ C_1 &= 64 (\log(L) - \log(M)) \\ C_2 &= 10 (\log(L) - \log(S)) \end{aligned} \tag{3.11}$$

Cette formule étant assez compliquée, une formule plus simple, proposée par Yang [26] sera également utilisée dans nos tests :

$$\begin{aligned} A &= \log(R) + \log(G) + \log(B) \\ C_1 &= \log(R) - \log(G) \\ C_2 &= \log(B) - \frac{\log(R) + \log(G)}{2} \end{aligned} \tag{3.12}$$

3.4 Systèmes perceptuels

3.4.1 Système $L^*a^*b^*$

Le but premier des systèmes perceptuels est de faire correspondre le plus fidèlement possible les écarts couleur dans l'espace à notre façon de les percevoir.

$L^*a^*b^*$ est un autre espace développé par la CIE. Cette dernière le considère comme l'espace de référence lorsqu'il s'agit de calculer des écarts couleur. La composante L^* met en évidence l'opposition des couleurs noir-blanc (information de luminance), a^*, celle des couleurs vert-rouge et b^*, celle des couleurs bleu-jaune. Il possède donc également l'atout de dissocier luminance et chrominance. En outre, on peut remarquer qu'il pourrait aisément être classé parmi les espaces antagonistes, mais le fait que la CIE l'ait conçu en gardant en tête l'aspect perceptuel des couleurs justifie qu'on le place dans cette catégorie.

La transformation de passage du XYZ vers $L^*a^*b^*$ n'est pas linéaire, ce qui la rend plus coûteuse en temps de calcul. Elle se présente comme la fonction continue suivante :

$$\begin{aligned} L^* &= \begin{cases} 116 * \left(\frac{Y}{Y_w}\right)^{\frac{1}{3}} - 16 & \text{si } \left(\frac{Y}{Y_w}\right) > 0.008856 \\ 903.3 * \left(\frac{Y}{Y_w}\right) & \text{si } \left(\frac{Y}{Y_w}\right) \leq 0.008856 \end{cases} \\ a^* &= 500 * \left(f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right)\right) \\ b^* &= 200 * \left(f\left(\frac{Y}{Y_w}\right) - f\left(\frac{Z}{Z_w}\right)\right) \end{aligned} \tag{3.13}$$

où

$$f(x) = \begin{cases} x^{\frac{1}{3}} & \text{si } x > 0.008856 \\ 7.787 * x + \frac{16}{116} & \text{si } x \leq 0.008856 \end{cases} \tag{3.14}$$

Dans cette formule, (X_w, Y_w, Z_w) sont les coordonnées X,Y et Z du blanc de référence considéré. Dans nos expériences, nous n'en tiendrons pas compte et considérerons que $(X_w, Y_w, Z_w) = (255, 255, 255)$. Nous ne pourrons donc pas tirer pleinement parti de ses possibilités puisqu'il dépend des conditions d'acquisition.

Il est à noter que la CIE a introduit plus tard un nouvel espace $L^*u^*v^*$ similaire au $L^*a^*b^*$. Nous ne nous y intéresserons pas compte tenu des faibles différences qu'il présente par rapport à son prédécesseur.

3.4.2 Systèmes géométriques

Il existe un certain nombre de systèmes (HSV, HCI, HSI, ...) qui se basent sur un partitionnement uniforme d'un espace couleur de type LTS (luminance - teinte - saturation). Leurs principales différences sont les unités et la dynamique des axes de couleur, celles-ci étant mineures, nous n'étudierons que le HSV. Ces représentations ont été conçues pour qu'on puisse repérer les couleurs à partir de coordonnées géométriques. Ainsi, à l'espace HSV correspond un cône hexagonal, comme on peut le voir à la figure 3.2.

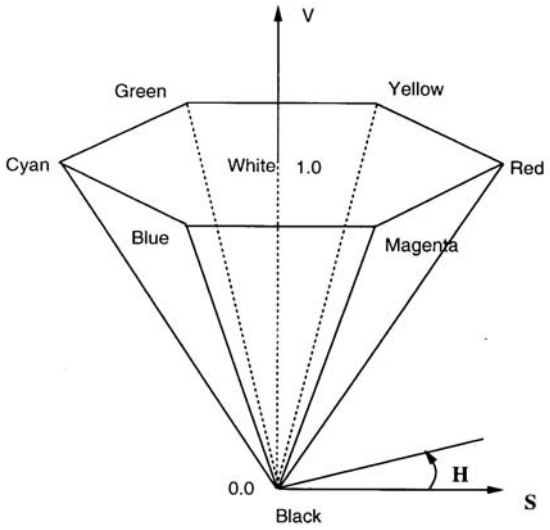


FIG. 3.2 – Représentation géométrique de l'espace HSV (source : [26])

Ce type de systèmes est très utile, car il dissocie des informations qui sont à la base du système de perception visuelle humaine. Dans le cas du HSV, les composantes d'une couleur décrivent directement la teinte (H), la saturation (S) et l'intensité (V).

Sur la figure, l'angle entre l'axe s et la droite H correspond à la teinte. La distance entre la pointe du cône et un point projeté sur l'axe v représente l'intensité. Enfin, la distance entre le centre de l'hexagone et un point, tous deux projetés sur un même plan horizontal, représente la saturation.

La conversion d'une image RGB vers le HSV ne s'effectue pas par un calcul simple ou matriciel, mais il s'agit plutôt d'un algorithme à appliquer à chaque pixel :

$V \leftarrow \max(R, G, B)$
`si` $\max(R, G, B) = 0$
    $H \leftarrow 0$
    $S \leftarrow 0$
`sinon`
    $S \leftarrow \left(\frac{\max(R,G,B)-\min(R,G,B)}{\max(R,G,B)}\right)$
    `si` $R = \max(R, G, B)$
        $H \leftarrow \left(\frac{G-B}{\max(R,G,B)-\min(R,G,B)}\right)$
    `sinon, si` $G = \max(R, G, B)$
        $H \leftarrow \left(2 + \frac{B-R}{\max(R,G,B)-\min(R,G,B)}\right)$
    `sinon`
        $H \leftarrow \left(4 + \frac{R-G}{\max(R,G,B)-\min(R,G,B)}\right)$
    `fin si`
`fin si`
$H \leftarrow (H * 60)$
`si` $H < 0$
    $H \leftarrow (H + 360)$
`fin si`

3.5 Systèmes d'axes indépendants

3.5.1 $X_1X_2X_3$

Comme expliqué plus tôt dans ce document, les systèmes d'axes indépendants permettent de travailler sur des images dont les composantes couleur sont décorrélées. Généralement, un espace pour lequel la corrélation a été ôtée est appelé $X_1X_2X_3$. Néanmoins, la conversion vers un système d'axes indépendants dépend de la distribution des couleurs considérée, ce qui signifie qu'elle diffère d'une image à l'autre. Cela donne donc lieu a une conversion trop coûteuse en temps de calcul, puisque les coefficients de la matrice à utiliser varient à chaque image. Nous ne nous attarderons donc pas sur cet espace.

Citons néanmoins la méthode de conversion la plus utilisée pour cet espace : la transformée de Karhunen-Loeve dont le détail peut être trouvé entre autres dans [24].

3.5.2 $I_1I_2I_3$

Cet espace tend à se confondre avec les espaces issus de la transformée de Karhunen-Loeve. Il est donc pratiquement décorrélé. De plus, comme nous allons le voir, la conversion vers l'espace $I_1I_2I_3$ nécessite peu de temps de calcul.

Il appartient aussi aux espaces de type luminance-chrominance : I_1 représente la luminance d'une couleur tandis que I_2 et I_3 en décrivent la chrominance.

Voici la formule de transformation utilisée :

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{-1}{2} \\ \frac{-1}{4} & \frac{1}{2} & \frac{-1}{4} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.15}$$

3.6 Etude comparative

3.6.1 Temps de conversion

Notre but étant de suivre une cible en temps réel, nous ne pouvons négliger le temps nécessaire à la conversion d'un espace à l'autre. Effectivement, même si nous trouvions un espace de couleur dans lequel l'algorithme du CHS est infaillible, nous ne pourrions l'utiliser si la conversion de l'image source dans cet espace se comptait en secondes.

Les tests ont été réalisés sous Matlab 7, en utilisant à chaque fois une implémentation « naïve », c'est-à-dire sans aucune optimisation. L'image utilisée a une taille de 320 pixels en largeur et 240 en hauteur. Chaque test a été réalisé 10 fois et le résultat affiché dans le tableau 3.1 correspond au temps moyen obtenu. La machine employée est un Pentium IV Xeon 2.4GHz (bi-processeur) avec 512Mo de RAM, tournant sous Windows XP. Toutes les formules détaillées dans les sections précédentes ont été soumises au test, ce qui signifie que certains espaces sont repris plusieurs fois dans le tableau des résultats (l'ordre d'apparition dans le tableau est identique à l'ordre d'introduction des formules).

Espace	Temps moyen (s)
CMY (1)	0.0186
CMY (2)	1.2889
XYZ	1.7199
YC_bC_r	1.6922
YIQ	1.6917
AC_1C_2 (1)	4.8456
AC_1C_2 (2)	1.5898
$L^*a^*b^*$	5.9686
HSV	0.1754
$I_1I_2I_3$	1.4911

TAB. 3.1 – Espaces de couleur - Temps de conversion

On remarque, en regardant le tableau 3.1, que la plupart des conversions nécessitent approximativement le même temps de calcul. Ceci s'explique par le fait qu'elles consistent généralement en une multiplication matricielle. On peut néanmoins espérer pour ces dernières une implémentation permettant le temps réel grâce à diverses optimisations.

Parmi les autres, certaines (CMY (1) et HSV) utilisent des opérateurs plus simples (soustractions pour le premier et algorithme peu complexe pour l'autre) et donc plus rapides tandis que d'autres (AC_1C_2 (1) et $L^*a^*b^*$) ne sont pas linéaires et se prêtent donc difficilement aux applications en temps réel. Ces deux derniers seront donc exclus dans la suite de ce travail.

3.6.2 Résultats du CHS

Après avoir décrit les principaux espaces de couleur et avoir testé les temps de conversion depuis l'espace RGB, nous allons maintenant comparer les performances du CHS sur des images converties au préalable dans chaque espace couleur présenté ci-dessus. En effet, il s'agit du critère le plus important pour le choix d'un espace : il faut qu'il améliore les performances du CHS par rapport au RGB standard. Pour ce faire, plusieurs séquences d'images ont été utilisées (disponibles en annexe A), en voici les caractéristiques :

1. déplacement d'un objet avec obstruction partielle et changement d'orientation (3 images).

2. changement d'éclairage sur un objet assez nuancé par rapport au reste de la scène (4 images).
3. changement d'éclairage sur un objet peu nuancé par rapport au reste de la scène (3 images).
4. déplacement d'un objet avec obstruction partielle (5 images).

Espace	**Quantification**	**Seq. 1**	**Seq. 2**	**Seq. 3**	**Seq. 4**
RGB	3-3-2	100	100	0	100
CMY (1)	3-3-2	100	100	0	100
CMY (2)	3-3-2	58	100	0	100
XYZ	3-2-3	33	75	25	100
YC_bC_r	0-4-4	92	100	75	100
YIQ	2-3-3	100	100	33	95
AC_1C_2 (1)	0-4-4	42	75	0	80
AC_1C_2 (2)	0-4-4	33	100	58	100
$L^*a^*b^*$	2-3-3	58	50	42	100
HSV	4-2-2	100	100	92	100
$I_1I_2I_3$	2-3-3	58	100	58	95

TAB. 3.2 – Espaces de couleur - Résultats du CHS

Les tests ont été réalisés sur la même machine que celle ayant servi à calculer les temps de conversion, avec Matlab 7 également. La taille des images est toujours de 320x240 pixels. A chaque fois, plusieurs quantifications ont été essayées et seuls les résultats liés à la meilleure d'entre elles sont affichés dans le tableau 3.2, les autres étant repris en annexe A.

Pour rappel, une quantification « $q_1 - q_2 - q_3$ » signifie que q_1 bits seront consacrés à la première composante couleur, q_2 à la seconde et q_3 à la dernière.

Chaque espace a reçu un score pour les images sur lesquelles le CHS a été effectué. Lorsque la cible n'était pas détectée, aucun point n'était attribué, tandis qu'un point était accordé pour une détection impeccable. Enfin, des cotes intermédiaires ont été également attribuées, respectivement 0.25 et 0.75, en cas de détection incorrecte mais peu éloignée de la position voulue et en cas de détection approximative (cible partiellement trouvée). Le tableau 3.2 reprend ces résultats exprimés en termes de pourcentages.

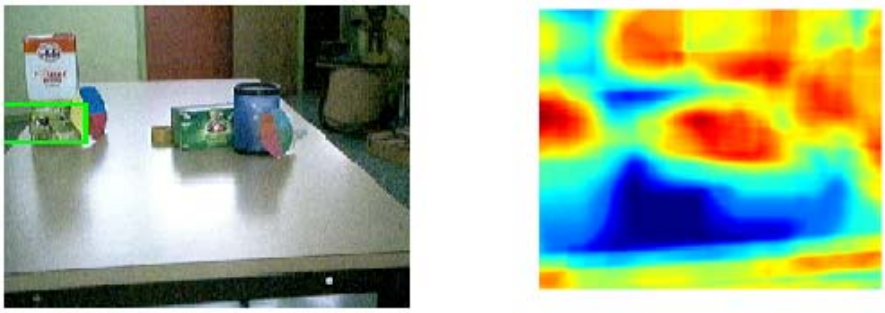
On remarque que, généralement, le RGB se prête assez bien au CHS. Ses résultats sont effectivement très corrects, à l'exception de la troisième séquence qui présente des conditions assez difficiles (la cible est une carte à puce de couleur proche de celle de la table sur laquelle elle se trouve).

Parmi les autres espaces performants, on retrouve le YC_bC_r et le HSV qui fournissent de meilleurs résultats que le RGB. Le HSV réalise d'ailleurs presque un sans-faute puisqu'il a trouvé la cible dans deux cas et il la localise partiellement dans la troisième image de la séquence 3. Cette supériorité peut s'expliquer par le fait qu'il dissocie les informations de chromacité et de luminance. Or, nous avons tenu compte de cela lors de la quantification afin de minimiser l'importance de la luminance et donc, mieux réagir aux changements de luminosité dans la séquence.

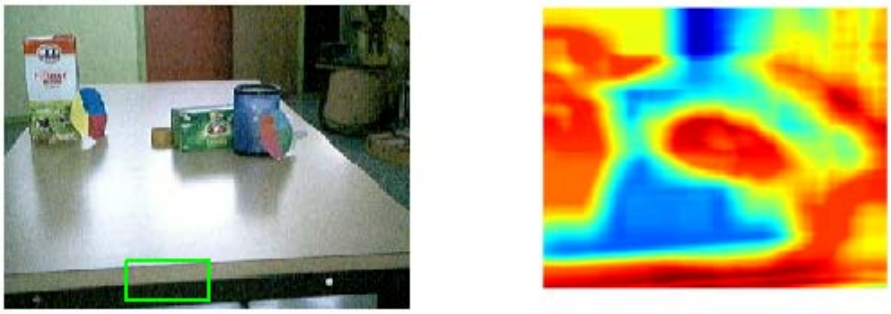
Enfin, on remarque qu'il y a trois espaces qui se montrent assez mauvais avec le CHS : le XYZ, le AC_1C_2 (première formule) et le $L^*a^*b^*$. Ainsi, ils échouent même dans la localisation de la cible pour certaines images des deux premières séries. La raison de ces échecs vient sans doute de la construction de ces espaces. En effet, ils ont été conçus selon une optique bien précise dont la quantification uniforme effectuée ne tient pas compte. Par exemple, l'espace RGB peut se représenter sous forme de cube qui se prête facilement à une quantification uniforme. Au contraire, d'autres espaces (comme le $L^*a^*b^*$) ont une représentation géométrique plus complexe. Ainsi, selon Tremeau et al. [24], il n'est pas conseillé d'employer le $L^*a^*b^*$ en dessous de 12 bits par composante. Nous n'en utilisons que 8 au total.

Nous allons aborder dans la suite de cette section les problèmes apparus lors de ces tests. Rappelons d'abord brièvement que, sur la surface de similarité, plus la couleur d'un pixel tend vers le rouge, plus la probabilité que ce pixel soit le coin supérieur gauche de la cible est élevée. Réciproquement, si la couleur tend vers le bleu, alors cette probabilité est faible.

Il faut remarquer à propos de ces résultats que le fait de les établir de manière numérique est tout à fait réducteur (approche « tout ou rien »). Ainsi, deux espaces couleur peuvent se voir affecter, pour une même image, un 0 pour la localisation d'une cible alors que leur échec peut être partiel. La figure 3.3 illustre ce problème. On y voit la deuxième image de la série 2 pour les espaces XYZ (quantification 3-2-3) et AC_1C_2 (deuxième formule, quantification 0-4-4), la surface de similarité produite par le CHS pour chacun



(a) Localisation de la cible et surface de similarité (XYZ 3-2-3)



(b) Localisation de la cible et surface de similarité (AC_1C_2 0-4-4)

FIG. 3.3 – Résultats quantitatifs réducteurs

d'entre eux est également affichée. Tous deux échouent au test, la cible étant la boîte de thé, au centre de la scène. Pourtant, nous comprenons mieux l'erreur du XYZ puisqu'une partie de la boite de lait présente une répartition des couleurs similaire à la cible.

Il faut également relativiser ces résultats vu que, parfois, l'algorithme peut localiser la cible avec précision, et donc obtenir le maximum de points, alors qu'une légère modification de l'image le pousserait à en situer l'emplacement à un tout autre endroit. Cela est illustré sur la figure 3.4 où on remarque sur la surface de similarité que le maximum ne se distingue pas clairement d'une grande partie de l'image.

Néanmoins, nous pouvons nous permettre d'ignorer les défauts que nous venons de citer. En effet, ces remarques concernent principalement les espaces sur lesquels le CHS a fourni de mauvais résultats. Parmi ceux-ci, aucun

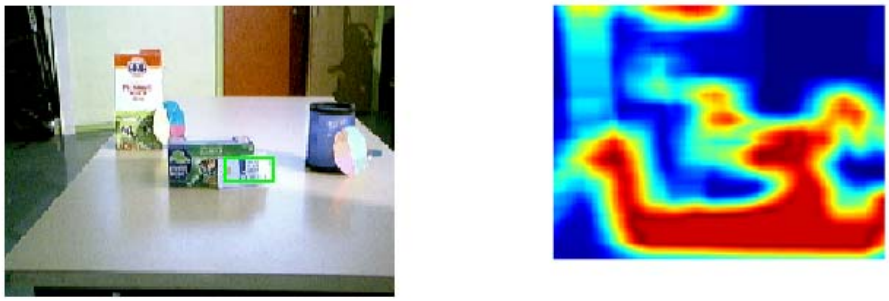


FIG. 3.4 – Non représentativité des résultats

ne justifie que nous modifions notre approche d'étude (leurs résultats sont souvent *très* mauvais).

Notre dernière remarque concerne la quantification. Certains espaces « symétriques » sont désavantagés par une quantification sur 8 bits, car on ne peut accorder autant d'importance à chaque composante. Par exemple, lors de l'utilisation de l'espace RGB, nous devons systématiquement défavoriser une composante (2 d'entre elles seront codées sur 3 bits tandis que la dernière le sera sur 2 bits). Dès lors, la qualité des résultats dépend de la quantification effectuée : une quantification 3-3-2 donnera de bons résultats sur des images utilisant peu le bleu, mais, sur une scène contenant de nombreuses variations de bleu, ils seront peu probants.

3.7 Conclusion

Comme évoqué précédemment, l'algorithme du CHS dépend fortement des histogrammes couleur. Or, à la base, les données exploitées étaient celles fournies directement par la caméra, à savoir des données codées en RGB. Nous nous sommes donc penchés sur d'autres formes de représentation de la couleur afin de tenter de trouver un espace qui améliorerait les résultats du CHS.

Pour ce faire, nous avons donc décrit brièvement les principaux espaces de couleurs en les classant en quatre catégories : les systèmes de primaires, les systèmes luminance-chrominance, les systèmes perceptuels et les systèmes d'axes indépendants.

Après en avoir survolé les caractéristiques ainsi que les formules de conver-

sion, nous avons effectué deux tests sur chacun d'entre eux. Il s'agissait de calculer le temps nécessaire à la conversion d'une image depuis l'espace RGB, espace utilisé par la « webcam » produisant les images, et chacun des espaces présentés auparavant. Ensuite, l'algorithme du CHS a été employé sur plusieurs séries d'images afin de voir quels étaient les espaces qui rendaient cet algorithme plus robuste.

Au vu des résultats repris précédemment dans ce chapitre, on pourrait envisager de retenir trois espaces de couleur pour la suite de ce travail :

- RGB
- YC_bC_r
- HSV

Le premier se comporte très bien dans de nombreux cas. Il commence cependant à montrer ses limites dans des cas plus extrêmes où les objets sont peu nuancés ou lors de changements de luminosité trop importants. Son plus gros atout se situe dans le fait qu'il est utilisé par la « webcam » et donc, qu'il ne demande aucun temps de conversion.

Le second, YC_bC_r, a également fait ses preuves lors des tests du CHS en montrant un comportement remarquable, même dans la troisième séquence qui posait problème pour la majorité des espaces couleur. Son seul inconvénient réside dans le temps nécessaire à la conversion, plus d'une seconde et demie. Or, ses performances étant en deçà de celles du HSV, nous ne l'utiliserons finalement pas dans les expériences ultérieures. Remarquons cependant que le temps de conversion peut sûrement être réduit en prenant en compte la quantification utilisée (0-4-4) qui ne nécessite pas de calculer la première composante (un gain de temps de 33% est envisageable).

Enfin, le HSV a donné les meilleurs résultats. Il a également l'avantage d'être peu coûteux en temps de calcul : moins d'un cinquième de seconde dans ces tests.

Quant aux autres espaces, les résultats obtenus étaient plutôt décevants. Cela ne signifie pas pour autant que ce soient de mauvais espaces. Le problème vient surtout d'une conversion inefficace. Ainsi, certains d'entre eux nécessitent la prise en compte de paramètres qui nous sont inconnus, tandis que d'autres, tels que le $L^*a^*b^*$, supportent mal la quantification uniforme que nous appliquons. Chaque espace a ses spécificités et les hypothèses que

nous avons fixées (temps réel, matériel standard, ...) ne nous permettent pas d'en tenir compte.

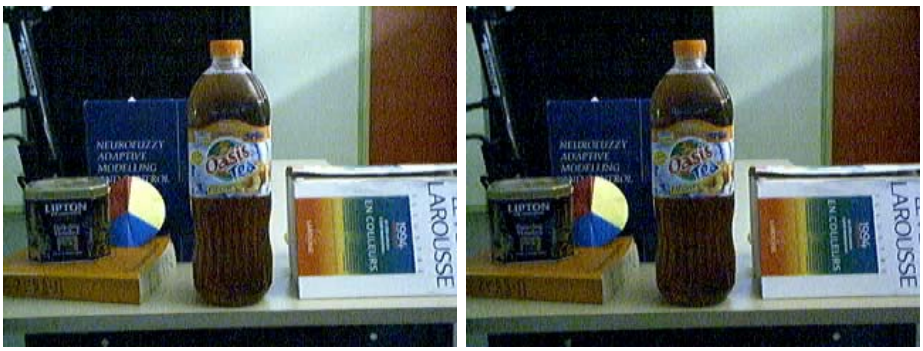
En conclusion, cette étude nous a permis de mettre en évidence deux espaces de représentation de la couleur, le RGB et le HSV. L'avantage de ce dernier est qu'il sépare les informations de luminance et de chrominance, ce qui lui permet de mieux supporter les variations de luminosité. Ce dernier point sera abordé plus en détails dans le chapitre 4.

Chapitre 4

Robustesse à l'illumination

4.1 Introduction

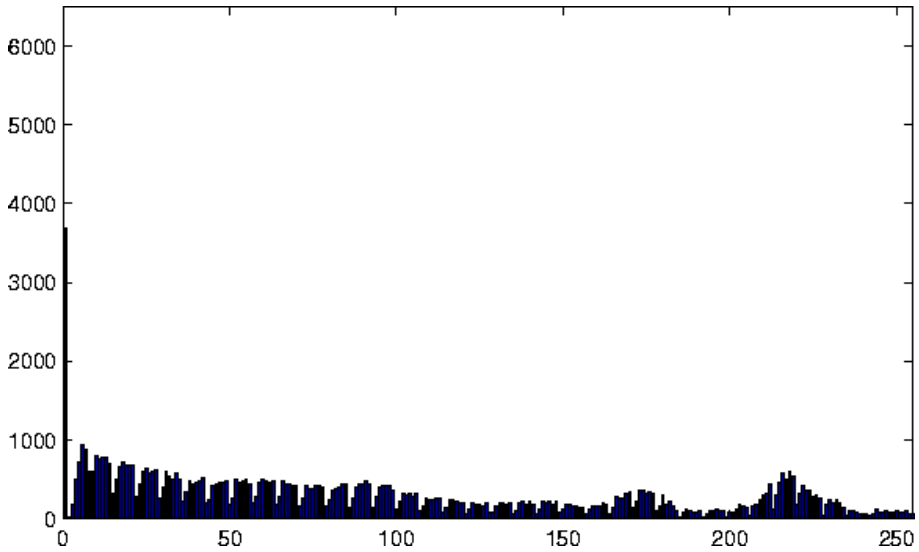
L'intersection d'histogrammes, même si elle a de nombreux avantages, reste très sensible à la luminosité de la scène. En effet, lors d'un changement de luminosité, les couleurs perçues d'un objet sont modifiées et engendrent des histogrammes différents. On remarque en général une translation des « bins » de l'histogramme (figure 4.2 pour les images de la figure 4.1).



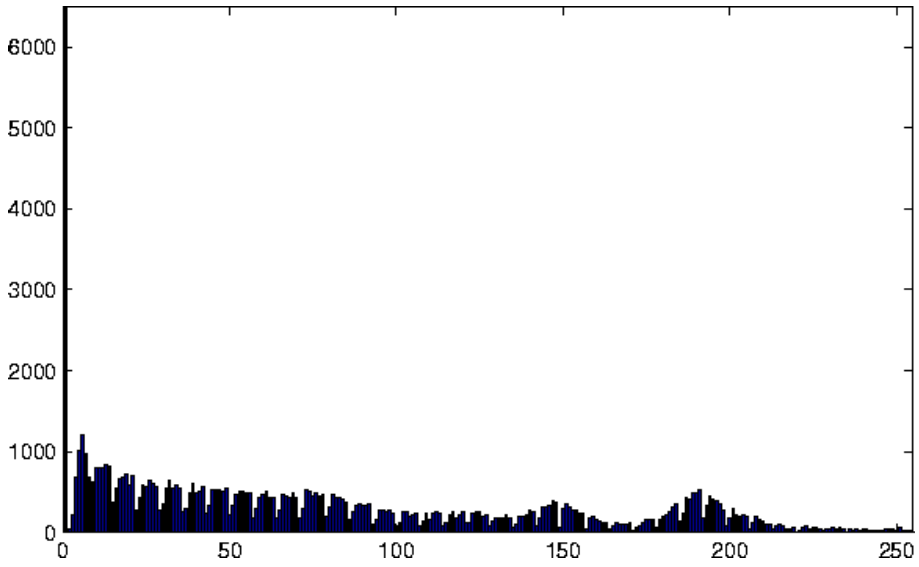
(a) Image n°1 plus claire (b) Image n°2 plus sombre

FIG. 4.1 – Images d'exemple

On peut voir sur cet exemple un déplacement général des « bins » vers la gauche (le pic autour du « bin » 220 se retrouve autour du « bin » 190) et une importance plus prononcée du « bin » 0 (passe de 3800 à 6200). Le « bin » 0 représente le noir, la couleur la plus sombre du cube RGB. Nous



(a) Histogramme de l'image n°1



(b) Histogramme de l'image n°2

FIG. 4.2 – Modification de l'histogramme

pouvons expliquer ce déplacement par un assombrissement général de la scène. L'ensemble des points dans le cube RGB représentant les pixels va donc se déplacer vers l'origine du cube, (0,0,0), qui est le noir absolu). Il va

donc y avoir une translation des « bins » représentant l'image vers le « bin » 0.

Ces changements dans l'histogramme ont pour conséquence de rendre l'intersection de très mauvaise qualité lorsque l'on compare un même objet perçu avec deux éclairages différents . On peut voir que, si un objet était défini par deux « bins » et qu'une légère translation était opérée (figure 4.3), l'intersection chuterait fortement alors qu'il s'agirait du même objet.

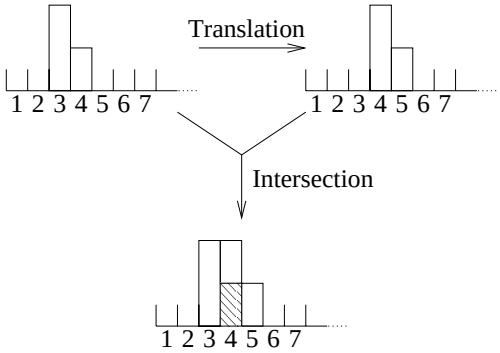


FIG. 4.3 – Exemple de translation dont l'intersection vaut 0.33

Ce phénomène peut être engendré aussi bien par la lumière ambiante que par une ombre mal placée. La luminosité de la scène peut être fortement différente de celle de la référence (par exemple, peu ou pas de lumière). Nous pouvons aussi trouver des scènes éclairées par des sources comme des spots ou des lampes halogènes. Ces derniers fournissent une lumière qui ne rayonne pas de façon équivalente sur toute la scène. Nous observons donc des apparitions d'ombres et de zones très éclairées (figure 4.4).

FIG. 4.4 – Exemple de scène avec une source halogène

Il est assez fréquent de ne pas pouvoir localiser la cible avec le CHS alors qu'elle apparaît clairement sur l'image. Nous pourrions simplement tester la luminosité de la scène et afficher son incapacité à trouver la cible si la luminosité n'est pas uniforme et suffisante. Bien qu'elle soit simple et résolve la plupart des cas litigieux, cette solution n'est pas acceptable et il est nécessaire de mettre en oeuvre des techniques qui permettent de pallier ce problème.

Les chapitres suivants développent trois techniques qui, chacune à leur manière, essaient d'améliorer le CHS. Il est important que chacune de ces techniques permette, d'une part, d'améliorer la valeur de l'intersection entre la cible de référence et la cible dans l'image, et d'autre part, de ne pas ou peu augmenter la valeur de l'intersection pour les autres zones de l'image analysée. Il faut, en effet, que ce que l'on ajoute à la méthode de base nous permette de mieux reconnaître la cible sans augmenter nos chances de la confondre avec des objets parasites. Nous établissons ainsi le critère de qualité de chaque technique.

4.2 Paramétrage du CHS

4.2.1 Introduction

Deux paramètres importants qui peuvent influencer la qualité de l'intersection du CHS sont l'espace couleur et sa quantification. En effet, ils sont à la base de la représentation des images comparées. Nous pouvons donc supposer que les choix réalisés sur ces deux paramètres peuvent améliorer ou affaiblir l'indépendance à l'illumination. Nous allons donc voir, pour chacun d'entre eux, les choix que l'on peut faire et si il y a une solution optimale au problème qui nous occupe.

4.2.2 Espace couleur

L'ensemble des espaces couleur ne seront pas repris ici puisqu'ils ont été étudiés en profondeur précédemment (chapitre 3). Nous allons plutôt nous poser la question de savoir ce qui définit un bon espace de couleur qui pallie au problème du changement d'illumination. Comme nous l'avons vu dans l'introduction, le problème se pose lorsqu'il y a un déplacement des pixels de la cible dans des « bins » voisins.

Lorsque l'on étudie les propriétés des espaces couleur, on peut noter un grand avantage aux espaces couleur, comme le HSV, qui séparent l'information de luminance et celle de chrominance. Cette caractéristique nous permet de quantifier différemment les composantes et ainsi créer des « bins » qui seront plus stricts sur l'information de chrominance mais plus larges sur l'information de luminance. Par exemple, le HSV est quantifié en général 4-2-2 (importance au H, la teinte). Dans le cas du RGB, il est impossible de faire cette quantification asymétrique puisque chacune de ses composantes a autant d'information de chrominance que de luminance.

Parmi les espaces retenus dans le chapitre 3, le HSV apporte cette dissociation d'information sur les couleurs. Les couleurs de la cible dans la scène seront donc plus probablement dans le même « bin » que les couleurs correspondantes de la référence. En effet, comme les « bins » sont plus larges sur les informations de luminance, ces couleurs peuvent être plus éloignées l'une de l'autre sous l'effet de l'éclairage sans changer de « bin ». La cible de référence sera donc plus localisable. Le choix de l'espace couleur peut aider à l'invariance de luminosité mais n'est pas une solution complète au problème. Les pixels changent, peut-être plus difficilement, de « bin » mais ils finissent néanmoins par le faire. Il est donc nécessaire de combiner un bon espace couleur avec d'autres méthodes.

4.2.3 Quantification

Il y a deux grands choix à faire sur la quantification d'un espace. Premièrement, la granularité définit la taille des sous-espaces. Appliquer une quantification 3-3-3 ou 5-5-5 donnera des résultats de CHS fort différents. Nous avons la possibilité de créer des sous-espaces grands (par exemple 3-3-3) ou des sous-espaces très fins (par exemple 5-5-5). Plus ces derniers seront étendus, moins l'effet d'un déplacement dans l'espace couleur aura d'influence puisqu'ils resteront dans le même « bin ». Malheureusement, l'effet pervers est de perdre toute précision dans la représentation de l'image. Nous aurions, dans le cas extrême, une quantification qui serait très robuste aux changements d'illumination mais qui serait incapable de faire la différence entre le bleu et le rouge. Dans le cas contraire, une quantification trop fine définit mieux les images mais devient très sensible à tout déplacement des couleurs. Il n'y a aucun « excellent » choix de quantification qui permette à la fois d'améliorer l'indépendance aux changements d'illumination et de garder une

qualité optimale du CHS.

Deuxièmement, comme vu dans la section précédente, nous pouvons quantifier indépendamment les différentes composantes. Dans les espaces qui le permettent, tels que le HSV, il est possible de faiblement quantifier les composantes fortement influencées par les changements de lumière. Cela permet d'augmenter l'indépendance tout en réduisant au minimum la qualité des informations sur l'image.

Nous pouvons aussi trouver des quantifications plus « exotiques ». Par exemple, il est possible de construire des quantifications qui tentent de scinder plus harmonieusement les différences de teinte. Sural et al. [22] propose de segmenter l'espace HSV de façon différente selon la saturation. Si la saturation est faible, la quantification se fait uniquement sur base de V sinon uniquement sur base de H. L'argumentation de cette méthode repose sur la répartition des couleurs dans l'espace HSV. Lorsque la saturation est basse, on se trouve près de l'axe central qui définit la gamme du blanc au noir selon la valeur de V. Dans le cas contraire, une certaine couleur définie par H est suffisamment présente et on se base uniquement sur le H. Intuitivement, cette segmentation semble cataloguer chaque couleur par sa teinte et devrait permettre une indépendance forte à la luminosité.

Malheureusement, après vérification expérimentale, on peut remarquer que nous perdons énormément d'information sur les images. Une cible composée de couleurs claires va être quantifiée dans les blancs (saturation faible) et ne pourra plus être reconnue lorsque l'image deviendra un peu plus sombre et que le bleu ressortira. Il n'y a plus non plus de graduation, ne serait-ce que faible, dans les saturations de couleur. Un objet vert foncé serait aussi bien reconnu que la cible comportant un vert très clair. Les résultats de ce type de segmentation n'ont pas été suffisamment concluants pour être utilisés par la suite.

4.2.4 Conclusion

Le choix de l'espace couleur et de sa quantification est d'une importance primordiale dans l'indépendance à l'illumination. Ils influencent de façon importante la qualité des solutions du CHS. Comme cela a été vu dans les parties précédentes, deux choix différents d'espace couleur et de quantification peuvent générer des solutions fort différentes. Certaines sont de très

mauvaise qualité. Il faut donc pouvoir faire des choix qui donnent des résultats de CHS satisfaisants tout en gardant un maximum de robustesse à l'illumination. Ces paramètres ne sont pas des solutions au problème mais sont des bases à sa résolution. L'espace HSV et une quantification 4-2-2 donnent les meilleurs résultats observés.

4.3 Algorithme « cross-bins »

4.3.1 Introduction

La position dans l'histogramme d'un pixel de couleur illuminé différemment n'est évidemment pas aléatoire. Les deux couleurs vont se retrouver dans des « bins » voisins dans l'espace couleur. Si un pixel bleu foncé de la référence se retrouve être bleu clair dans l'image analysée, les deux pixels, représentés dans l'espace couleur, seront dans le voisinage l'un de l'autre. Si la distance est trop grande, ils seront comptabilisés dans des « bins » différents mais proches. Afin de mieux réagir à ce problème, nous pourrions prendre en compte les « bins » de « couleur voisine » dans l'intersection. Cette intervention des « bins » voisins transforme l'algorithme « bin-par-bin » en un algorithme « cross-bins ».

Sur la figure 4.5, trois types généraux d'intersection de deux histogrammes translatés sont illustrés. Le premier représente une méthode « bin-par-bin » comme le CHS, méthode ignorant tout des « bins » de couleur voisine. Le deuxième est une méthode « cross-bins »qui compare chaque « bin » avec tous les autres. Le dernier compare les « bins » de couleur qui se ressemblent sans se borner au même « bin ». La troisième méthode est la plus intéressante. En effet, la deuxième prend bien en compte les couleurs dans les autres « bins » mais compare des couleurs beaucoup trop éloignées, qui ne se correspondent en rien.

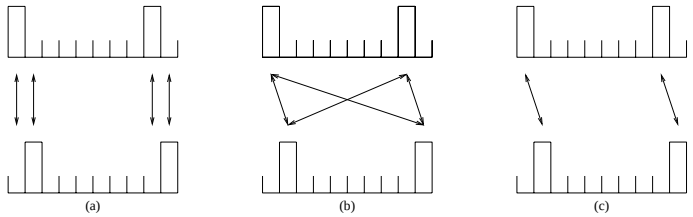


FIG. 4.5 – Principe général des méthodes « cross-bins »

Différentes solutions ont déjà été proposées dans ce sens (« quadratic-form distance », « match distance », « Kolmogorov-Smirnov distance », . . .). Une des techniques les plus utilisées grâce à sa flexibilité est l'« Earth Mover's Distance ».

4.3.2 « Earth Mover's Distance »

Principe de l'EMD

L' « Earth Mover's Distance » [21] (EMD) est une méthode pour comparer deux distributions se basant sur une « distance de base » entre deux éléments dans l'espace de ces distributions. Elle est réalisée en calculant le coût minimal à payer pour transformer une distribution en une autre. Le calcul de ce coût revient au calcul de la solution d'un problème bien connu de programmation linéaire, le problème du transport.

Ce dernier représente le problème de transport de biens entre plusieurs entrepôts, chacun ayant un certain nombre de biens, et plusieurs clients, ces derniers ayant chacun une certaine capacité. Pour chaque couple entrepôt-client, un certain coût de transport lui est lié. Résoudre ce problème revient à trouver le flux de biens à un coût minimal qui satisfasse la demande des clients (figure 4.6).

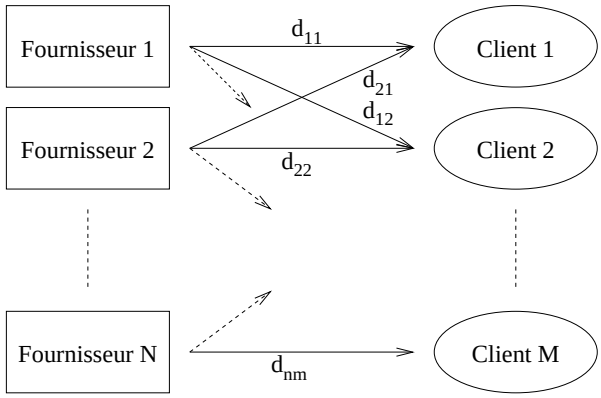


FIG. 4.6 – Schémas du problème du transport

La transition entre ce problème initial et le problème qui nous occupe est assez simple. Il suffit de définir une des deux distributions de pixels comme un ensemble d'entrepôts avec un stock de biens (pixels) et l'autre distribution

comme un ensemble de clients avec un besoin en biens (pixels). Le coût de transport des pixels est défini par la « distance de base ». La distance qui sera utilisée plus bas sera la distance L_1 dans l'espace couleur considéré.

La technique de comparaison d'images couleur proposée par Rubner et al. [21] n'utilise pas les histogrammes conventionnels. Comme vu précédemment, un histogramme est une correspondance entre un ensemble de vecteurs entiers à d-dimensions dans l'ensemble des réels non-négatifs. Chaque vecteur représente les « bins » dans un partitionnement de l'espace couleur et leur réel associé, la masse de la distribution qui tombe dans le « bin » correspondant. Ce partitionnement est malheureusement identique pour chacune des images. Si la quantification est trop grossière, une image fortement colorée sera représentée par l'histogramme avec trop peu d'information. Inversement, si la quantification est trop forte, certaines images avec peu de diversité de couleurs donneront un histogramme avec énormément de « bins » vides rendant les calculs ultérieurs plus complexes que nécessaires. Un histogramme ne peut donc être à la fois précis et efficace.

Pour pallier ce problème, les images sont quantifiées par une méthode de « clustering » qui crée un ensemble de « bins » qui recouvre efficacement la distribution tout en n'étant pas trop grand pour inclure dans le même « bin » des couleurs perçues différemment par l'oeil. Le résultat de ce « clustering » est la signature $\{s_j = \{m_j, w_j\}\}$ où chaque « cluster » s_j est défini par sa moyenne m_j (vecteur à d-dimensions) et par son poids w_j. Les signatures ne peuvent plus être comparées « bin par bin » puisqu'il n'y a plus de correspondance directe. Comme nous allons le voir, l'EMD prend en compte le centre de chacun des « bins » pour réaliser ses comparaisons.

A partir de tous ces concepts, nous pouvons définir formellement le principe de l'EMD comme suit.
Soit $P = \{(p_1, w_{p_1}), \cdots, (p_m, w_{p_m})\}$ la première signature avec m « bins » où p_i est le représentant du « bin » et w_{p_i} est le poids du « bin ».
Soit $Q = \{(q_1, w_{q_1}), \cdots, (q_n, w_{q_n})\}$ la seconde signature avec n « bins ».
Soit $D = [d_{ij}]$ la matrice de distance de base où d_{ij} est la distance de base entre les « bins » p_i et q_j.
Nous essayons de trouver $F = [f_{ij}]$, où f_{ij} correspond au flux entre p_i et q_j, qui minimise :

$$TRAVAIL(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{4.1}$$

Avec les contraintes :

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n \tag{4.2}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \qquad 1 \leq i \leq m \tag{4.3}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_j} \qquad 1 \leq j \leq n \tag{4.4}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min \left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j} \right) \tag{4.5}$$

La première contrainte s'assure que les pixels sont déplacés de P à Q et pas inversement. La seconde contrainte limite le nombre de pixels à déplacer à partir des « clusters » de P à leurs poids. La troisième contrainte assure que les « clusters » de Q ne recevront pas plus de pixels que leurs poids. Enfin, la quatrième contrainte maximise le mouvement de pixels entre les deux signatures. Pour la dernière contrainte et le problème qui nous occupe, le minimum peut être omis puisque nous comparons toujours des images avec un nombre égal de pixels. Une fois le flux optimal F trouvé, on définit l'EMD comme étant le travail normalisé par le flux optimal :

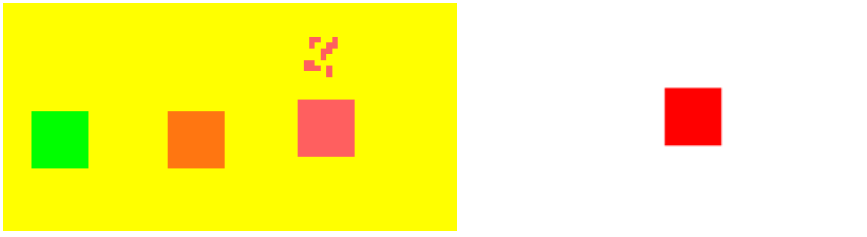
$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{4.6}$$

Cette normalisation est nécessaire car les deux signatures peuvent ne pas être de la même taille et le facteur qui normalise est égal à la masse totale de la plus petite des deux signatures grâce à la quatrième contrainte. La matrice de distance d_{ij} est définie selon l'espace couleur sous-jacent.

Cette technique a été développée premièrement pour l'« image retrieval » (recherche d'images similaires dans une base de données) où le calcul des signatures peut être réalisé à l'avance. Seule la signature de l'image recherchée doit être calculée avant de réaliser les comparaisons. Malheureusement, dans la problématique qui nous concerne, les images à traiter arrivent au fur et à mesure et le temps de réaliser une technique de « clustering » sur l'ensemble des images à considérer est prohibitif si nous voulons du temps réel. Nous avons donc appliqué l'EMD à partir d'une quantification normale. La transition de l'histogramme en une signature se fait naturellement en définissant l'élément de la signature s_j avec, comme m_j, le centre du « bin » j et avec, comme w_j, la valeur de h_j. Une différence significative avec cette approche, comme nous le verrons plus loin, est la différence de taille entre une signature réalisée par une technique de « clustering » et celle dérivée d'un histogramme.

Validation de la méthode

Un test synthétique a été réalisé afin de démontrer les intérêts de l'EMD lors de la recherche d'une cible dans une image. Le test a été réalisé avec une quantification HSV(4,2,2) sur une cible de couleur rouge pur et une image de recherche comportant deux candidats avec des couleurs sensiblement différentes (figure 4.7).



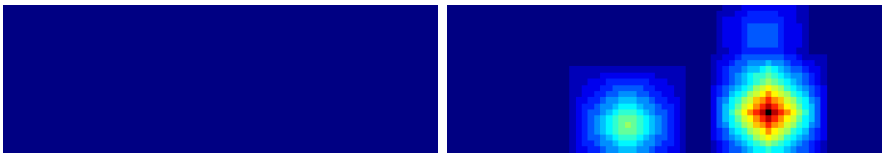
(a) Image synthétique (b) Cible rouge pure (0,255,255)

FIG. 4.7 – Test synthétique de validation

La matrice de distance de l'EMD a été définie comme suit :
Soit (dH,dS,dV) la différence quantifiée de (H,S,V) entre le représentant du « bin » i et du « bin » j, alors pour chaque couple (i,j) :

$$\begin{aligned} &\text{Si} \quad dH < 2 \\ &\qquad d_{ij} = dH * 10 + dS + dV \\ &\text{Sinon} \\ &\qquad d_{ij} = 20 \end{aligned}$$

Cette matrice définit un voisinage pour chacun des « bins » qui se dégrade très vite sur l'axe des H et très peu sur S et V car la teinte est l'indicateur principal de changement de couleurs perçues. La valeur 20 définie aléatoirement grande correspond à une très mauvaise correspondance entre ces deux « bins », une trop grande dissimilarité. Le 20 permet de transporter des pixels vers des « bins » qui ne leur sont pas chromatiquement voisins ($dH \geq 2$) en dernier recours puisque l'algorithme tente de minimiser les coûts. Les surfaces de similarité du CHS et de l'EMD montrent que le CHS est incapable de reconnaître un des carrés candidats malgré leur forte ressemblance avec la cible. L'EMD, en contre-partie, retrouve les deux carrés et de façon plus forte celui dont la teinte est plus ou moins équivalente (figure 4.8).



(a) Surface du CHS (b) Surface de l'EMD

FIG. 4.8 – Résultats du test synthétique

Le test synthétique permet de valider le fonctionnement attendu de l'EMD mais ne permet pas d'apprécier l'avantage d'utiliser l'EMD sur un simple CHS. A cette fin, nous avons besoin de lancer des tests sur des images de la caméra et comparer les surfaces de similarité de l'EMD et du CHS. Nous avons tenté de réaliser ces tests mais, bien que l'ensemble du code nécessaire à réaliser ces comparaisons était prêt, nous n'avons pu les terminer. En effet, l'utilisation d'une technique de type « cross-bins » induit une augmentation de la quantification afin de pouvoir définir un voisinage qui ne s'étend pas

trop dans l'espace couleur (il est inutile d'inclure le bleu dans le voisinage du rouge). Cette augmentation du « binning » rend le temps de calcul de l'EMD complètement démesuré. Le traitement d'une simple image peut prendre jusqu'à une semaine. Cette limitation technique ne permet donc pas de faire une comparaison conséquente. De plus, quelle que soit la qualité des résultats de l'EMD, un temps de calcul pareil ne peut évidemment pas être accepté pour du traitement en temps réel.

Il nous faut donc une simplification de l'EMD ou une autre approche afin de garder l'avantage du « cross-bins » tout en gardant un temps de calcul raisonnable. Un autre algorithme, utilisé en automatique, va permettre d'approcher ce problème sans le travail massif nécessaire pour l'EMD. Le « Cerebellar Model Articulation Controller » (CMAC) va nous permettre de définir un moyen de comparer des images tout en gardant l'idée de voisinage que nous explorons dans cette partie.

4.3.3 « Cerebellar Model Articulation Controller »

Le « Cerebellar Model Articulation Controller »ou CMAC est un algorithme de type réseau neuronal qui modèle, de façon simple, le fonctionnement du cervelet. Le cervelet est la partie du cerveau qui effectue la régulation des activités musculaires du corps. Il reçoit une large quantité d'information qui lui permet de donner aux programmes moteurs du mouvement une organisation chronologique et spatiale. Le fonctionnement de cette partie du cerveau est relativement bien connu, ce qui a permis à Albus [1] en 1975 de définir l'algorithme CMAC.

Cet algorithme permet, à partir d'un certain nombre d'entrées (les stimuli), de définir des paramètres de sortie (la réponse) qui ont été appris. Cette association entre les entrées et les sorties a de nombreux avantages. Premièrement, elle est extrêmement rapide. Deuxièmement, elle est très robuste aux bruits dont peuvent souffrir les entrées, ce qui lui permet de toujours pouvoir donner les sorties voulues. Et finalement, sa capacité d'apprentissage lui permet d'être très flexible. Grâce à tous ces avantages, le CMAC a souvent été utilisé (et étendu) dans le cadre d'applications de contrôle robotique adaptatif temps-réel.

Algorithme CMAC

L'algorithme du CMAC peut être défini de deux manières différentes, soit comme un réseau de neurones, soit comme une table d'associations (« look-up table »). Nous allons uniquement développer la seconde méthode qui correspond à ce que nous allons utiliser pour le CHS.

Son fonctionnement général peut être découpé en trois parties :

1. quantification de l'espace d'entrée
   permet de réduire la taille de l'espace d'entrée
2. généralisations locales du vecteur quantifié
   construit plusieurs quantifications plus grossières qui généralisent le vecteur d'entrée
3. somme pondérée des résultats de la généralisation
   calcule le résultat à partir d'une table de poids

Avant de voir plus en détail ces trois étapes, nous allons présenter un exemple simple du CMAC afin de mieux comprendre son fonctionnement. Soit un espace à 1 dimension, le vecteur d'entrée y est quantifié entre 0 et $N-1$. Nous définissons trois généralisations de l'entrée qui sont représentées par des couches. Chaque couche est représentée par une requantification de y plus grossière et une translation des classes d'une couche par rapport à l'autre. Chaque y, compris entre 0 et $N-1$, est transformé en un triplet (μ_1, μ_2, μ_3) dont chaque élément est compris entre 0 et $\frac{N-1}{3}$ (figure 4.9). Le résultat final, x, est donné par la somme des poids correspondant au triplet.

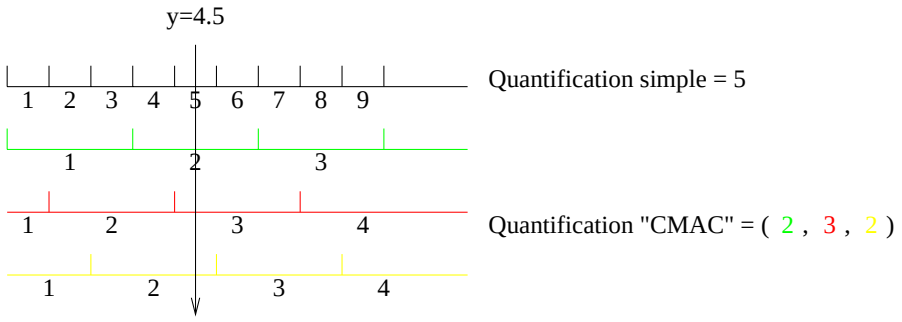


FIG. 4.9 – Exemple des deux premières étapes du CMAC

Nous pouvons définir de façon plus générale et plus formelle l'algorithme comme suit. En premier lieu, l'algorithme reçoit n_y entrées. Chaque entrée (y_i) doit être comprise entre min_i et max_i qui sont connus. Nous quantifions chacun des y_i dans l'ensemble des entiers de 0 à $res_i - 1$ où res_i a été défini préalablement comme la résolution de l'entrée y_i. En résumé, la première étape se définit par :

$$\text{Entrées} : y_1, \cdots, y_{n_y}$$
$$\text{Sorties} : q_1, \cdots, q_{n_y}$$

$$\begin{aligned}
&\text{Pour chaque } i = 1, \cdots, n_y \\
&\quad \text{si } y_i < min_i \text{ alors } y_i = min_i \\
&\quad \text{si } y_i > max_i \text{ alors } y_i = max_i \\
&\quad q_i = \lfloor res_i \frac{y_i - min_i}{max_i - min_i} \rfloor \\
&\quad \text{si } q_i \geq res_i \text{ alors } q_i = res_i - 1
\end{aligned}$$

Nous avons donc maintenant un ensemble q_i d'entrées quantifiées qui va être traité par n_a « unités d'association » (UA). Chaque UA applique un certain déplacement d_{ij} à chaque variable et requantifie le tout. L'ensemble des d_{ij} est défini entre 0 et $n_a - 1$ puisque un déplacement de n_a donnerait la même couche qu'un déplacement de 0. Il en résulte un vecteur μ à n_a-dimensions obtenu par :

$$\text{Entrées} : q_1, \cdots, q_{n_y}$$
$$\text{Sorties} : \mu_1, \cdots, \mu_{n_a}$$

$$\begin{aligned}
&\text{Pour chaque } j = 1, \cdots, n_a \\
&\quad \text{Pour chaque } i = 1, \cdots, n_y : p_{ij} = \left\lfloor \frac{q_i + d_{ij}}{n_a} \right\rfloor \\
&\quad \mu_j = \text{CMACINDEX}(j, p_{1j}, p_{2j}, \cdots, p_{n_y j})
\end{aligned}$$

La fonction CMACINDEX retrouve l'index du sous-espace considéré dans la couche à partir des index de chacun de ses axes. Cette fonction, pour μ_n, est de la forme :

$$\sum_{j=1}^{n_y} \left(p_{nj} \cdot \prod_{i=1}^{j-1} \left(\left\lfloor \frac{res_i - 2}{n_a} \right\rfloor + 2 \right) \right)$$

où les éléments multipliés sont les résolutions de p_{ij} que l'on obtient en ajoutant 1 au maximum de p_{ij}, lui même défini par :

$$\begin{aligned}\max(p_{ij}) &= \left\lfloor \frac{\max(q_i) + \max(d_{ij})}{n_a} \right\rfloor \\ &= \left\lfloor \frac{(res_i - 1) + (n_a - 1)}{n_a} \right\rfloor \\ &= \left\lfloor \frac{res_i - 2}{n_a} \right\rfloor + 1\end{aligned}$$

Le vecteur μ donne les index dans une table de poids W définie ou apprise par l'algorithme d'apprentissage supervisé du CMAC. Soit n_x le nombre de sorties, le résultat final est construit à partir de cette table de poids :

Entrées : $\mu_1, \cdots, \mu_{n_a}$

Sorties : $x_1, \cdots, x_{n_x}$

Pour chaque $k = 1..n_x$

$x_k = 0$

Pour chaque $j = 1, \cdots, n_a : x_k = x_k + W_{k\mu_j}$

Cette approche a un gros problème : la taille de W. En effet, nous pouvons calculer la taille de la matrice W par :

$$\begin{aligned}\text{Taille de } W &= n_a . \prod_{i=1}^{n_y} (\max(p_{ij}) + 1) \\ &\quad n_a . \prod_{i=1}^{n_y} \left(\left\lfloor \frac{res_i - 2}{n_a} \right\rfloor + 2 \right)\end{aligned}$$

Pour simplifier, si res_i est suffisamment grand et est le même pour toutes les entrées alors :

$$\begin{aligned}\max(p_{ij}) &\simeq \frac{res}{n_a} \\ \text{Taille de } W &\simeq n_a \left(\frac{res}{n_a} \right)^{n_y} \\ &= \frac{res^{n_y}}{n_a^{n_y - 1}}\end{aligned}$$

Nous pouvons voir que, même dans un exemple ridiculement petit où $res = 256$, $n_a = 6$ et $n_y = 4$, le nombre de poids s'élève à 2 millions. Pour pallier ce problème, la fonction CMACINDEX est remplacée par une fonction de « hashing » CMACHASH qui renvoie un index dans une table beaucoup plus réduite de poids.

Nous n'utiliserons pas l'entièreté de cet algorithme afin de remplacer l'EMD. Le but voulu est différent du contrôle robotique. La partie importante que nous allons utiliser pour explorer une méthode « cross-bins » est constituée par les deux premières étapes de l'algorithme : quantification et généralisation locale. A partir de là, quelques adaptations sont nécessaires.

Algorithme CMAC adapté au CHS

L'algorithme du CHS amélioré par l'algorithme du CMAC se déroule comme suit. Tout d'abord, nous quantifions les images comme pour le CHS normal. Ensuite, nous calculons la « quantification CMAC » de chacun des pixels. Cette dernière est directement tirée des valeurs sortantes des unités d'association (vecteurs p_{ij}). A partir de ces valeurs, nous construisons n_a histogrammes qui représentent les distributions des pixels dans les couches. Finalement, nous réalisons une intersection normale couche par couche entre la cible et les images à comparer. La valeur finale est la moyenne des intersections. En résumé, l'algorithme construit chaque « histogramme CMAC » :

Entrées : les T pixels de l'image

Sorties : $hist_1, \cdots, hist_{n_a}$

Pour chaque pixel $y = (y_1, y_2, y_3)$ de l'image

Pour chaque i=1..3

$q_i = \lfloor res_i \frac{y_i}{255} \rfloor$

si $q_i \geq res_i$ alors $q_i = res_i - 1$

Pour chaque $j = 1, \cdots, n_a$

$p_{ij} = \left\lfloor \frac{q_i + d_{ij}}{n_a} \right\rfloor$

$hist_{jp_{ij}} = hist_{jp_{ij}} + 1$

L'intersection de deux de ces histogrammes se fait simplement par :

Entrées : $hist_1, \cdots, hist_{n_a}$ et $hist'_1, \cdots, hist'_{n_a}$
Sortie : z

$z = 0$
Pour chaque $i = 1, \cdots, n_a$
  $zh = 0$
  Pour chaque $j = 1, \cdots, th$
    $zh = zh + \min(hist_{ij}, hist'_{ij})$
  $z = z + \frac{zh}{T}$
$z = \frac{z}{n_a}$

où th représente le nombre de « bins » dans une couche.

La méthode d'intersection par CMAC va permettre de créer une intersection graduelle entre des pixels de « bins » voisins. En effet, une des propriétés du CMAC est de ne changer que d'une seule valeur d'association lorsque l'on se déplace dans un sous-espace voisin. Nous aurons donc, au fur et à mesure que les pixels s'éloignent du « bin » d'origine, une intersection qui deviendra de plus en plus faible. Nous pouvons voir sur la figure 4.10 une coupe de l'intersection d'un pixel de couleur (128,128,128) avec toutes les autres couleurs du cube RGB. La quantification normale est de 5-5-5 et il y a 7 couches. Nous remarquons, comme prévu, une intersection forte des pixels très proches et une diminution graduelle jusqu'à devenir nulle.

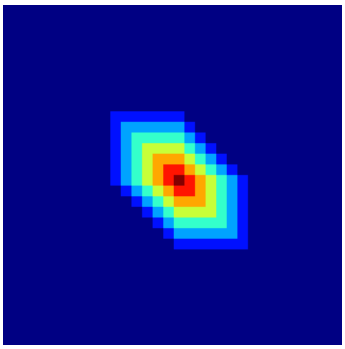


FIG. 4.10 – Intersection CMAC (bleu=0 et rouge=1)

Nous avons donc une méthode « cross-bins » pour comparer deux images et qui, comme nous le verrons plus bas, va beaucoup plus vite que l'EMD. Par rapport à l'EMD, nous pouvons voir que cet algorithme est un cas spécial, un type de matrice EMD précis. En définissant la matrice EMD pour chaque « bin » i et j comme suit :
Soit P_i le vecteur des quantifications CMAC du « bin » i
Soit P_j le vecteur des quantifications CMAC du « bin » j
Alors,

$$
\begin{aligned}
&D_{ij} = 0 \\
&\text{Pour chaque } k = 1, \cdots, n_a \\
&\qquad \text{Si } P_{ik} = P_{jk} \\
&\qquad\qquad D_{ij} = D_{ij} + \frac{1}{n_a}
\end{aligned}
$$

Nous obtenons une matrice qui fonctionne de la même manière que le CHS avec le CMAC. Notre nouvel algorithme n'est donc qu'un sous-ensemble des possibilités données par l'EMD. C'est une restriction on ne peut plus acceptable. En effet, la plupart des matrices définissables n'ont aucun intérêt pour la comparaison d'images puisque seules les couleurs voisines doivent être comparées. Le CMAC fonctionne de manière similaire à l'EMD pour la classe de problèmes qui nous intéresse. Sur le papier, le CMAC semble être très intéressant. Nous allons voir maintenant comment il se débrouille en pratique.

Paramétrage et résultats pratiques

Il y a trois grands paramètres à déterminer qui influent fortement sur le résultat de l'algorithme :

1. les translations,
2. le nombre de couches,
3. la quantification.

Il faut d'abord savoir comment décider des translations des différentes couches du CMAC. Dans l'exemple de la figure 4.9, il n'y avait qu'une seule entrée et le seul décalage possible était de décaler de 1 chaque couche par

rapport à la précédente. Lorsqu'il y a trois entrées (comme dans notre problème, les trois composantes de couleur), les possibilités de décalages sont de $(n_a)^3$. Nous avons, dans ce cas, beaucoup plus de possibilités que de décalages à choisir. La solution la plus simple est de décaler chaque composante de 1 comme dans l'exemple de l'entrée unique. Le résultat de ce décalage a déjà été montré sur la figure 4.10. On peut remarquer que le voisinage n'est pas très bien réparti autour du pixel comparé. Parks et Militzer [17] proposent une stratégie de décalage qui crée un bien meilleur voisinage. On peut voir que lorsque cette stratégie est utilisée avec les même paramètres que l'exemple précédent, le voisinage est mieux réparti (figure 4.11). Nous avons utilisé cette méthode avec tous les tests sur le CMAC.

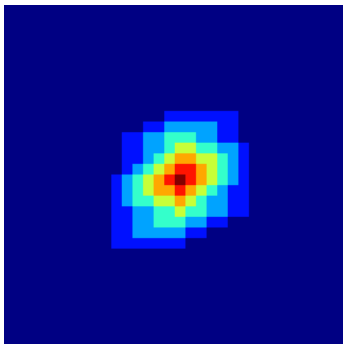


FIG. 4.11 – Intersection CMAC (bleu=0 et rouge=1)

Nous avons, ensuite, le problème fortement lié du nombre de couches et de la quantification. Ils vont ensemble définir la taille du voisinage d'un « bin ». Il faut que ce voisinage ne soit ni trop grand ni trop petit. De plus, le temps de calcul est totalement dépendant de ces deux facteurs. D'une part, augmenter la quantification augmente la taille des histogrammes et donc le temps de calcul des intersections. D'autre part, augmenter le nombre de couches augmente le nombre d'histogrammes mais réduit la taille de ceux-ci. Après de nombreux tests, nous avons établi que les valeurs qui donnaient les meilleurs résultats tout en satisfaisant nos contraintes de temps, étaient une quantification de 5-5-5 (en RGB) et de 3 à 5 couches. Le temps de calcul oscille entre 20 et 30 ms pour le traitement complet d'une image.

Les tests réalisés n'ont malheureusement pas donné de résultats très concluants. En effet, en comparant le CHS simple et le CHS avec CMAC, il est apparu que, pour toute quantification du CHS simple, il y a moyen

de paramétrer le CHS avec CMAC pour donner les mêmes résultats à une différence moyenne de 0.02. Par exemple, pour un CHS avec une quantification 3-3-3 (RGB), un CHS avec CMAC 5-5-5 et 4 couches réagit de la même manière.

Nous avons uniquement constaté une meilleure qualité des surfaces qui sont en général plus régulières (grâce au voisinage graduel du CMAC). L'apport du CMAC ajoute un peu de précision dans l'intersection sans pour autant avoir des avantages extraordinaires. Un trop grand ou trop petit voisinage de CMAC a les mêmes problèmes qu'une quantification du CHS trop grande ou trop petite.

4.3.4 Conclusion

Bien que l'idée d'utiliser les informations des « bins » voisins dans l'intersection semble très bonne, il faut remarquer qu'une application simple d'un voisinage ne semble pas donner de biens meilleurs résultats que le CHS avec quantification fixe. Néanmoins, cette piste de solution ne devrait pas être écartée car la translation des pixels dans des « bins » voisins est un fait. Les « bins » voisins peuvent donc être une mine d'informations pour l'image à comparer.

4.4 Algorithme à référence dynamique

4.4.1 Introduction

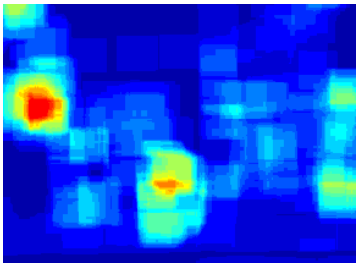
Jusqu'à présent, la recherche du CHS a toujours été effectuée à partir d'une image de référence, constante, de l'objet recherché. En général, cette image était prise sous de bonnes conditions de lumière afin de bien dissocier les différentes couleurs de la cible. Nous avons discuté maintes fois des conséquences de cette pratique sur l'intersection du CHS lorsque la lumière de la scène est différente de la luminosité de la référence. Il serait donc intéressant de modifier la référence au fur et à mesure du suivi afin de prendre en compte la différence de luminosité entre la référence d'origine et la cible courante dans l'image analysée.

4.4.2 Pistes de solution

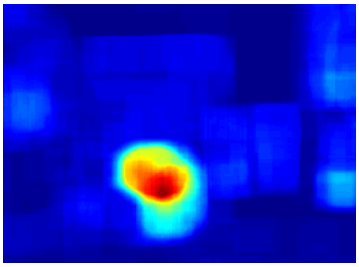
L'idée première est de pouvoir adapter l'histogramme de référence afin qu'il corresponde à la luminosité de la scène. Une solution serait, premièrement, de calculer la luminosité moyenne de la scène et celle de la référence. Ensuite, nous appliquons une translation à l'histogramme afin qu'il ait la même luminosité moyenne. Un exemple est donné dans la figure 4.12 où l'on recherche la cible à trois couleurs. Sans adaptation de la cible, le maximum de la surface est de 0.08 et est incorrect. La même recherche CHS avec une adaptation du CHS permet de retrouver la cible avec un maximum de 0.42.

(a) Image et solution sans adaptation

(b) Image et solution avec adaptation



(c) Surface sans adaptation



(d) Surface avec adaptation

FIG. 4.12 – Comparaison de CHS avec et sans adaptation de la cible

Cette technique simple améliore l'intersection du CHS dans de nombreux cas mais fonctionne très mal dans d'autres. En effet, elle ne marche que si la scène est illuminée de façon uniforme. Si l'objet recherché se trouve dans

une ombre ou dans un rayon de lumière, sa luminosité locale peut être fort différente de la luminosité moyenne de la scène. Nous pouvons donc avoir un effet pervers de la translation où celle-ci rend plus difficile la localisation de la cible.

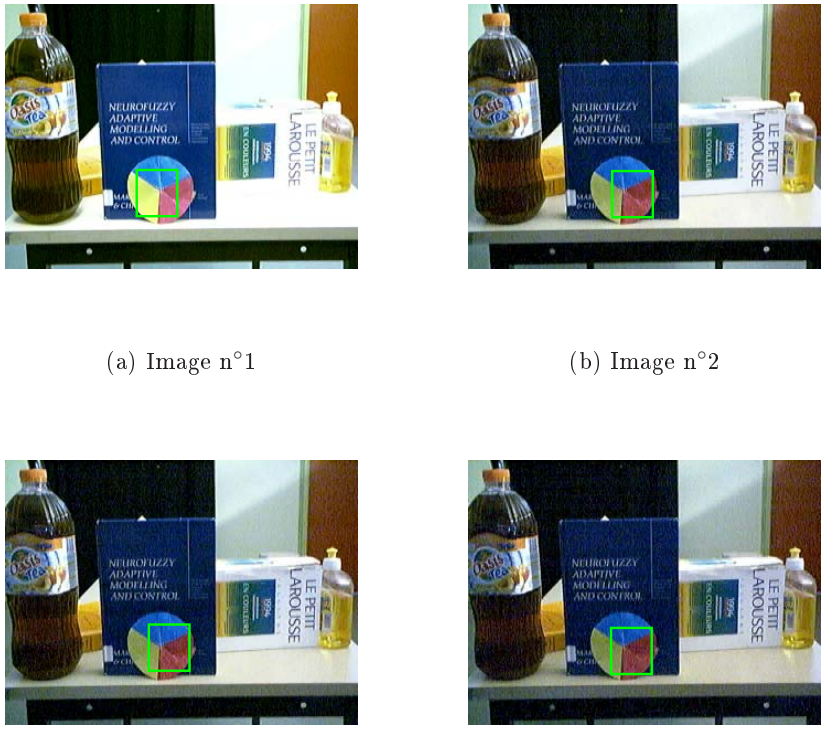
Pour pallier cette difficulté, il suffit de n'utiliser que la luminosité de la zone où se trouve la cible dans l'image. Il se pose le problème de localiser une première fois la cible sans adaptation de la référence. En contre-partie, si la cible est localisée une première fois, nous avons une information qui permet de faire le suivi de manière optimale. Il serait même plus intéressant de recréer un histogramme à partir de cette solution afin de remplacer celui de référence. Nous avons ainsi une indépendance à une illumination différente sur la surface même de la cible dans l'image. Ce remplacement peut être réalisé à chaque itération du suivi et ainsi s'adapter en permanence à la cible en mouvement.

Cependant, le remplacement complet de la référence est dangereux pour deux raisons. Premièrement, si la première localisation est erronée, nous perdons toute information sur notre cible. Deuxièmement, si la cible dans l'image est plus sombre, nous perdons une information de mélange de couleurs qui nous serait utile si la cible revient dans une zone plus éclairée.

4.4.3 Historique de références

La solution considérée est de constituer un « historique » de références. Cet historique de taille t est construit au fur et à mesure des analyses d'images. La première image, la référence fixe, est gardée en tout temps. Les $t-1$ autres sont remplacées au fur et à mesure par les nouvelles solutions du CHS. La solution finale du CHS est déterminée par le maximum des t surfaces de similarité générées par les t références. Le balayage de l'image restant le même pour chacune des surfaces, nous n'augmentons le travail à réaliser que par $t-1$ intersections en plus. Ce surcoût nous force à utiliser un facteur t relativement petit, de l'ordre de 2 ou 3.

Un exemple montre l'avantage de l'historique dans le suivi d'une cible tricolore avec un historique de taille 3 (figure 4.13). Le suivi est correct et l'assurance de trouver la cible est beaucoup plus grande dès que l'historique est construit (tableau 4.1, et images n°3 et n°4 de la figure 4.13).



(a) Image n°1 (b) Image n°2

(c) Image n°3 (d) Image n°4

FIG. 4.13 – Suivi avec historique de références

Image	max référence	max historique
Image 1	0.85	0.85
Image 2	0.17	0.17
Image 3	0.1	0.68
Image 4	0.06	0.6

TAB. 4.1 – Résultats du suivi par historique

L'historique de références fonctionne très bien à deux conditions. Il faut d'abord être sûr que la solution du CHS représente bien la cible. Dans le cas contraire, nous allons polluer l'historique et complètement corrompre notre suivi. Ce problème peut être réglé grâce aux indicateurs, comme cela sera vu dans le chapitre 6. La deuxième condition est d'avoir un minimum de pixels parasites dans la solution. Ces derniers peuvent apparaître lorsque la

cible a une taille ou une orientation différente à notre canevas. Un exemple de dysfonctionnement est présenté sur la figure 4.14. La cible, l'étiquette de jus d'orange, est peu à peu pervertie pour finalement ne plus être trouvée. La solution de la première itération a beaucoup trop de pixels parasites et pervertit l'historique. Les solutions suivantes s'éloignent peu à peu de l'étiquette pour finalement s'arrêter sur le mur blanc. La seule solution est de compléter cet historique avec des méthodes d'adaptation à la taille (comme cela sera vu au chapitre 5) et à l'orientation.

FIG. 4.14 – Perte de la cible à cause de l'historique

4.4.4 Conclusion

L'apport de la dynamique de la référence est un atout indéniable dans le suivi. Elle permet de s'adapter en continu aux changements d'illumination de la cible. Cependant, cette adaptation n'est correcte que si elle comprend tous les facteurs, à savoir l'orientation, la taille et l'illumination. Nous n'avons donc proposé ici qu'une solution partielle du problème mais qui fonctionne relativement bien lorsque les conditions sont respectées.

4.5 Conclusion

Nous nous sommes attachés tout au long de ce chapitre au problème épineux de la robustesse à l'illumination du CHS. Nous avons présenté les conséquences sur le CHS et les solutions possibles que nous avons étudiées.

Il est presque impossible de trouver une solution « magique » pour être complètement indépendant de l'illumination. Elle a un rôle si important dans la perception des couleurs que, dans une méthode telle que le CHS, son impact peut être dramatique.

Nous avons cependant tenté d'améliorer la localisation d'un objet dans l'image sans imposer une quelconque illumination de la scène. Parmi les solutions proposées, le choix de l'espace couleur et la quantification sont des facteurs qui peuvent être établis pour limiter le problème. La création, tout au long du suivi, d'un historique de références n'est pas parfaite mais est une piste de solution si elle est complétée avec d'autres méthodes d'adaptation. Finalement, la solution des algorithmes « cross-bins » qui semblaient être la plus prometteuse donne des résultats décevants. La résolution de ce problème est donc loin d'être clôturée mais certains éléments ont été apportés et plusieurs pistes de solutions restent à développer.

Chapitre 5

Estimation de la taille

5.1 Introduction

Un des problèmes majeurs lié à la détection d'une cible consiste à estimer la taille de cette dernière dans l'image. Effectivement, avant de procéder à la localisation d'un objet, on en crée une image de référence et on en calcule la signature. Cette dernière est ensuite utilisée tout le long de la séquence pour repérer la position de l'objet recherché. Cependant, puisqu'on considère le problème de la localisation dans le cadre de l'asservissement visuel d'un robot, on ne peut négliger le fait que la taille de la cible varie au cours du temps.

En effet, la cible occupe une partie croissante de la scène au fur et à mesure que le robot se rapproche de l'objet qu'il doit saisir. Rapidement, l'objet à localiser aura trop peu de similitudes avec la cible définie à l'origine que pour pouvoir être détecté correctement.

Dès lors, il est nécessaire de modifier l'algorithme du CHS afin de prendre en compte les changements de taille. Cette amélioration, présentée dans [9], sera expliquée en détail dans la section 5.2.

Néanmoins, nous verrons que cet apport à la méthode du CHS ne suffit pas. En effet, grâce à ce dernier, nous pouvons localiser un objet en considérant qu'il est plus petit ou plus grand, mais le facteur de variation de taille doit encore être calculé. Il nous faut donc élaborer un algorithme autour du CHS qui tirera parti de cette possibilité de redimensionnement. Des recherches ont été réalisées et deux méthodes, présentées dans [8], seront explicitées dans les sections 5.3 et 5.4 :

– l'analyse d'intervalle,
– le « Colour Histogram Footprint ».

Ces deux méthodes permettent d'évaluer la taille actuelle de la cible par rapport à la taille d'origine. Pour chacune d'entre elles, nous détaillerons le fonctionnement, puis nous effectuerons une batterie de tests pour déterminer leurs performances avant de conclure avec les perspectives qu'elles offrent.

5.2 Gestion des redimensionnements avec le CHS

Comme expliqué ci-dessus, il est crucial de pouvoir faire une estimation de la taille de l'objet que l'on cherche à localiser sur l'image. Pour définir un taux de redimensionnement r, le TROP propose de se baser sur la relation suivante [9] :

$$\frac{S}{S_{ref}} = \frac{d_{ref}^2}{d^2} \tag{5.1}$$

où

– S est la surface apparente de la cible (nombre de pixels) dans l'image analysée.
– S_{ref} est la surface de la cible dans l'image de référence.
– d et d_{ref} représentent la distance entre la caméra et l'objet, respectivement, dans l'image analysée et dans l'image de référence.

A partir de cette relation, on définit

$$r = \sqrt{\frac{S}{S_{ref}}} = \frac{d_{ref}}{d} \tag{5.2}$$

où r est le facteur de redimensionnement de la cible dans l'image analysée par rapport à sa taille dans l'image de référence.

Le paramètre que nous venons de définir doit ensuite être correctement pris en compte à différentes étapes de l'algorithme du CHS afin d'être utile :

– modification de la taille de la fenêtre de recherche,
– définition de l'histogramme de la cible.

Dans la version initiale du CHS, si on considère que la cible est de taille $l_1 \times l_2$ alors, la fenêtre de recherche est de taille $l_1 \times l_2$. Désormais, pour tenir compte de ce facteur de redimensionnement, il est nécessaire d'utiliser une fenêtre de taille $(r \times l_1) \times (r \times l_2)$.

De plus, le nombre de pixels qui entrent en compte dans le calcul de l'histogramme n'est plus le même lorsqu'on fait face à un changement de taille. Ainsi, il nous faut adapter l'histogramme de la cible de référence. Pour ce faire, on considère que les proportions de couleurs dans la cible redimensionnée sont identiques :

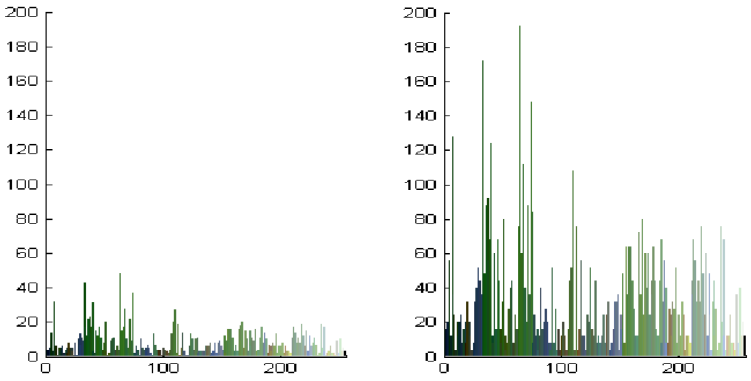
$$H_r = r^2 * H \tag{5.3}$$

où

- H_r est l'histogramme de la cible modifié en fonction du rapport r.
- H est l'histogramme de la cible dans l'image de référence.

La figure 5.1 illustre ce que nous venons d'expliquer sur les histogrammes. Ainsi, les images (a) et (b) sont identiques à un facteur d'échelle $r = 2$ près. L'histogramme (c) est donc quatre fois ($r^2 = 2^2 = 4$) plus petit que celui représenté en (d).

(a) Image originale (b) Image redimensionnée



(c) Histogramme de (a) (d) Histogramme de (b)

FIG. 5.1 – Redimensionnement d'image et d'histogramme

Ces deux modifications nous permettent donc d'employer une fenêtre de recherche contenant $(r \times l_1) \times (r \times l_2)$ pixels et d'en comparer l'histogramme avec H_r qui contient le même nombre de pixels. Il ne reste donc plus qu'à trouver une estimation du paramètre r pour pouvoir localiser avec précision la cible. Pour ce faire, plusieurs méthodes peuvent être imaginées, nous en verrons quelques unes dans les sections 5.3 et 5.4.

5.3 Analyse d'intervalle

5.3.1 Description

L'analyse d'intervalle consiste à évaluer les résultats obtenus par le CHS pour un certain nombre de valeurs du paramètre r. C'est une étape préalable à la mise au point d'un algorithme de recherche de la valeur de r. Ainsi, il faut vérifier que la fonction d'évaluation du CHS présente un maximum ainsi qu'une certaine monotonie. Si cela s'avère vrai, alors il est envisageable de développer une technique tirant parti de ce fait (par recherche dichotomique par exemple).

Il s'agit d'une méthode itérative. L'itération s'effectue sur le paramètre de redimensionnement r que prend en compte la nouvelle version du CHS. Il nous faut donc définir un intervalle $[a, b]$ et un pas de progression p, ce qui nous donne l'ensemble des valeurs de r pour lesquelles le CHS sera effectué :

$$r \in \{a + k * p \mid 0 \leq k \leq \frac{(b-a)}{p}\} \tag{5.4}$$

Chaque itération renvoie comme résultat une surface de similarité dont nous extrayons le maximum, noté $M(r)$, comme dans l'utilisation habituelle du CHS. La méthode de l'analyse d'intervalle part de l'hypothèse que la similarité est maximale lorsque la taille de la fenêtre de recherche correspond à la taille du modèle dans l'image. En d'autres termes, si R est le facteur de taille correct entre l'image de référence de la cible et la cible dans l'image à analyser :

$$R = \arg\max_r M(r) \tag{5.5}$$

L'idée d'itérer sur un ensemble de valeurs afin d'estimer correctement la taille d'une image n'est pas nouvelle. En effet, Cohen [11] a utilisé le même principe, à quelques différences près. La plus importante était que

l'algorithme qui donnait la probabilité était l'« Earth Mover's Distance » (EMD). Les résultats qu'il a obtenus étaient, selon lui, plutôt satisfaisants.

Buessler et al. [9], en présentant cette méthode, ont réalisé des tests sur une séquence d'images, avec deux cibles différentes, afin d'en vérifier le bien-fondé. Les images utilisées, ainsi que les objets à localiser, sont disponibles en annexe B. Leurs résultats étaient également satisfaisants comme le montre le tableau 5.1 (les deux séquences sont identiques, seule la cible change). On peut cependant remarquer que la taille est systématiquement sous-estimée.

Image		**Séquence *RCX***	**Séquence *FlunchBot***
	Taux réel R	**Taux estimé** r	**Taux estimé** r
Modèle	1.00	1.00	1.00
N°1	0.31	0.16	0.21
N°2	0.49	0.33	0.46
N°3	0.70	0.60	0.53
N°4	0.98	0.91	0.86
N°5	1.28	1.19	1.20

TAB. 5.1 – Estimation du taux de redimensionnement (source : [9])

Il est très intéressant d'étudier le problème de sous-estimation de plus près. Ce phénomène est dû au redimensionnement de l'histogramme de référence que nous effectuons. En effet, lors de cette opération, nous multiplions chaque « bin » par r^2, mais dans la réalité, des détails peuvent apparaître ou disparaître lors du déplacement, avec comme conséquence directe une modification de la distribution des couleurs. Prenons l'exemple synthétique de la figure 5.2. Sur cette dernière, nous nous sommes rapprochés de la cible

(a) Cible originale (b) Cible rapprochée

FIG. 5.2 – Apparition de détails lors du rapprochement de la cible

originale ($R = 5/3$) et une ligne jaune, invisible auparavant, est apparue. Ainsi, comme l'indique le tableau 5.2, le contenu de l'histogramme projeté

par notre calcul de redimensionnement ($H_r = r^2 H$) ne correspond pas à la cible lorsque nous la voyons de plus près.

Cible	**« Bin » rouge**	**« Bin » bleu**	**« Bin » jaune**
Initiale	3276	324	0
Projetée	9100	900	0
Rapprochée	9050	900	50

TAB. 5.2 – Contenu de l'histogramme

Ces détails peuvent être considérés comme du bruit, car ils modifient la distribution des couleurs de façon imprévisible et perturbent la reconnaissance de l'objet. Or, chaque augmentation de la taille de la fenêtre de recherche (chaque incrément de r) a pour conséquence la prise en compte d'un plus grand nombre de pixels de bruit. Dès lors, l'algorithme du CHS trouve r où $r < R$. Cette différence reste cependant relativement réduite.

Ci-après, nous allons voir par le biais de quelques tests que d'autres complications surviennent lorsqu'on désire obtenir le taux de redimensionnement correct. Pour cela, nous inspecterons, pour les valeurs de r, l'intervalle $[0.2, 2]$ avec un pas de 0.01, ce qui nécessitera donc 180 itérations de l'algorithme du CHS.

Lorsque nous présenterons un graphique de résultats, sauf indication contraire, il s'agira d'une représentation de maximum de similarité en fonction du facteur d'échelle employé, $M(r)$.

5.3.2 Tests sur images synthétiques

Nous utilisons ici des images très peu complexes dans lesquelles peu de couleurs sont présentes. La cible est un rectangle rouge dans lequel est incrusté un rectangle bleu. Chaque élément de la séquence contient cette même cible, la taille variant à chaque fois, avec du bruit ayant la couleur du fond, c'est-à-dire du jaune. La figure 5.3 illustre l'objet de référence ainsi qu'une image de la séquence (la séquence complète ainsi que les résultats sont disponibles en annexe B). Il y a en tout cinq images dans la séquence avec un facteur de redimensionnement r valant respectivement 0.40, 0.65, 1.00, 1.45 et 1.75.

Le premier test sur lequel nous allons nous pencher est le premier élément

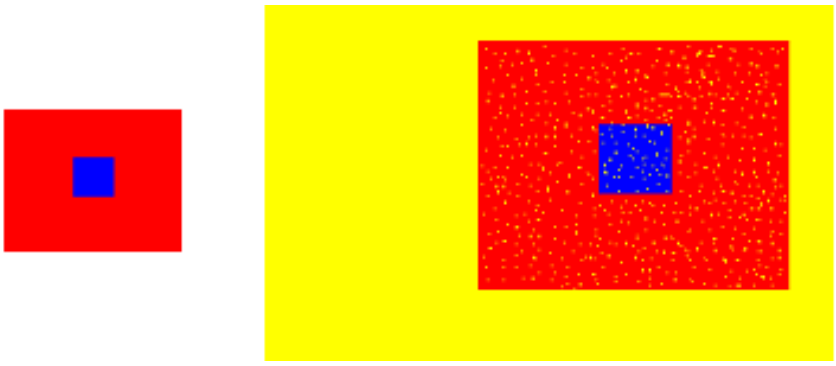


FIG. 5.3 – Cible utilisée sur la séquence (à gauche) et image n°4 (à droite)

de la séquence, ce qui signifie donc que la cible est plus petite sur notre image que sur celle de référence (le facteur d'échelle correct R est 0.4). La figure 5.4 reprend les résultats obtenus. Pour les valeurs de $r \in]0.5, 2.0]$, $M(r)$ continue à décroître exponentiellement. Comme nous ne nous intéressons pas à cette partie du graphique, nous l'en avons supprimée.

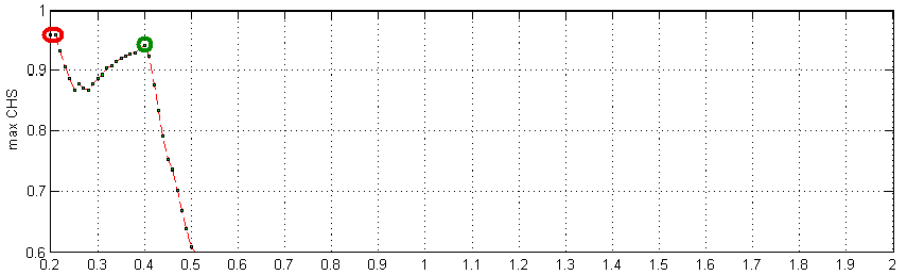


FIG. 5.4 – Résultat de l'analyse d'intervalle — $R = 0.4$

En regardant le graphique, on constate que, si on se contente de prendre le plus grand maximum de similarité, c'est-à-dire $R_{est} = \arg\max_r M(r)$, le facteur de redimensionnement estimé est de 0.2, ce qui n'est pas du tout correct. On remarque que $M(r)$ décroît pour $r \in [0.20, 0.28]$ (si on ne tient pas compte du léger pic en $r = 0.26$) avant d'augmenter jusqu'à atteindre un pic en 0.4, puis la courbe diminue de façon exponentielle. Le problème se situe clairement au début de la courbe où les valeurs de $M(r)$ sont supérieures à celle que nous recherchons, 0.4.

Le mauvais fonctionnement de la méthode d'analyse d'intervalle est montré de façon plus marquée sur la figure 5.5 qui contient le résultat de l'opé-

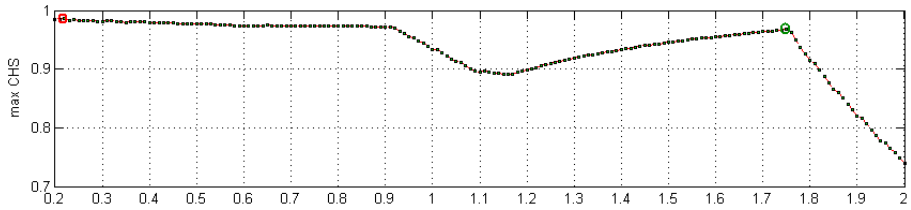


FIG. 5.5 – Résultat de l'analyse d'intervalle — $R = 1.75$

ration sur la cinquième image de la séquence. Ainsi, la zone problématique concerne ici l'ensemble des $r \in [0.20, 0.92]$, zone beaucoup plus grande que dans l'exemple précédent. Or, dans ce cas, le taux de redimensionnement correct est $R = 1.75$.

L'inefficacité de la méthode est due à deux facteurs :
- les images-tests présentées sont trop synthétiques ;
- des taux de redimensionnement trop petits impliquent l'utilisation d'une fenêtre de recherche de seulement quelques pixels, ce qui entraîne un mauvais fonctionnement de l'algorithme du CHS.

La première cause est uniquement due à l'utilisation de cibles synthétiques dans nos tests. En effet, si on considère les deux images de la figure 5.6, on remarque aisément qu'elles possèdent toutes deux le même histogramme. Seule la répartition spatiale des couleurs diffère. Grâce à cet exemple, on

FIG. 5.6 – Images différentes possédant un même histogramme

comprend directement que le CHS fournirait d'excellents résultats pour n'importe quelle petite taille, la cible diminuée étant alors localisée près de la bordure du rectangle bleu. En d'autres termes, la présence de symétries dans l'image de gauche implique qu'une demi-image ait, proportionnellement, le même histogramme que l'image globale. Or, la distribution du bruit rend très probable le fait de trouver une sous-image avec un meilleur CHS que celui du rectangle pour $r = R$. Néanmoins, cet inconvénient est uniquement dû aux images que nous avons employées, il y a peu de chances qu'il apparaisse

lors de l'utilisation de vraies séquences.

La seconde raison du dysfonctionnement de la méthode de l'analyse d'intervalle vient du fait que nous utilisons des valeurs très petites pour le paramètre r. Or, celui-ci a une influence directe sur la taille de la fenêtre de recherche servant à balayer l'image à analyser. Une valeur trop peu élevée réduirait donc la fenêtre à quelques pixels. Ainsi, si la cible de référence fait 50×20 pixels et que nous définissons $r = 0.2$, cela nous donne 10×4 comme dimensions de la fenêtre de recherche. En d'autres termes, sa surface couvre $1/25^{ème}$ de la surface originale (40 pixels, à la place de 1000). Dès lors, l'algorithme du CHS a tendance à localiser l'objet à un endroit où il n'est pas. Effectivement, puisque l'histogramme est réduit à quelques valeurs (la plupart des « bins » étant vides), une zone de l'image pour laquelle les couleurs correspondent à celles de la cible réduite est plus facile à trouver.

Dans le cas des images synthétiques, il est bien évident que la cause du problème rencontré est celle que nous avons expliquée en premier ci-dessus, autrement dit une trop grande simplicité des images utilisées. Nous verrons dans la section suivante ce qui concerne les séquences d'images concrètes.

Pour éviter le problème lié aux faibles valeurs de r, la possibilité que nous avons envisagée est de modifier la définition de R. Pour rappel, ce dernier était défini comme suit :

$$R = \arg\max_{r} M(r) \tag{5.6}$$

Désormais, nous considérerons que l'estimation correcte du taux de redimensionnement R est la suivante :

$$\begin{cases} M(R-p) \leq M(R) \\ M(R+p) < M(R) \\ \nexists\ R_2\ tel\ que \begin{cases} M(R_2-p) \leq M(R_2) \\ M(R_2+p) < M(R_2) \\ R < R_2 \end{cases} \end{cases} \tag{5.7}$$

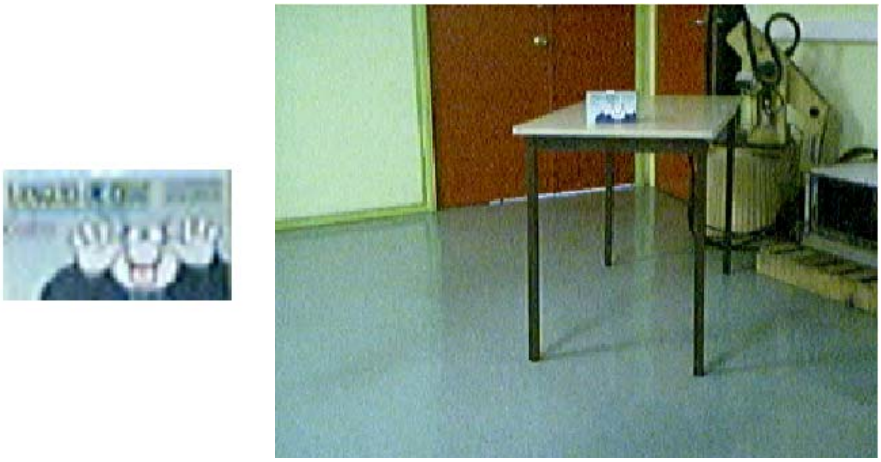
où p correspond au pas de progression dans l'intervalle.

En d'autres termes, il s'agit de trouver, sur la courbe des $M(r)$, un maximum local et ce, en effectuant l'analyse en partant de la borne supérieure de l'intervalle et en décroissant vers la borne inférieure (sur le graphique, de droite à gauche).

Cette nouvelle définition nous convient parfaitement pour les tests sur les images synthétiques. En effet, la modification que nous y avons apportée permet de détecter à chaque fois correctement la taille de la cible dans l'image. Nous allons donc maintenant analyser les résultats obtenus lors des tests sur des images réelles.

5.3.3 Tests sur images réelles

Dans cette section, nous allons nous pencher sur les résultats que donne la méthode de l'analyse d'intervalle sur des images réelles. La séquence utilisée comprend quatre images sur lesquelles la taille de la cible, une boîte de chocolat, décroît. La figure 5.7 illustre l'objet de référence ainsi qu'une image de la séquence. La séquence complète est disponible en annexe B. Les valeurs du facteur de redimensionnement r, calculées empiriquement, sont respectivement 1.70, 1.00, 0.56 et 0.44.



(a) Cible (b) Image n°3

FIG. 5.7 – Cible utilisée sur la séquence et image n°3

Comme nous l'avons expliqué dans la section précédente, l'utilisation d'une fenêtre de recherche trop petite pousse l'algorithme à tirer des mauvaises conclusions sur l'emplacement de la cible. Dans cette séquence, la taille de la boîte de chocolat utilisée comme référence est de 49x28 pixels. Des taux de redimensionnement autour de 0.2 ou 0.3 réduiront donc cette fenêtre à

quelques dizaines de pixels, ce qui fera apparaître le problème expliqué plus haut.

Analysons maintenant le résultat de la méthode de l'analyse d'intervalle sur la troisième image de la séquence, où le taux recherché vaut 0.56. Comme

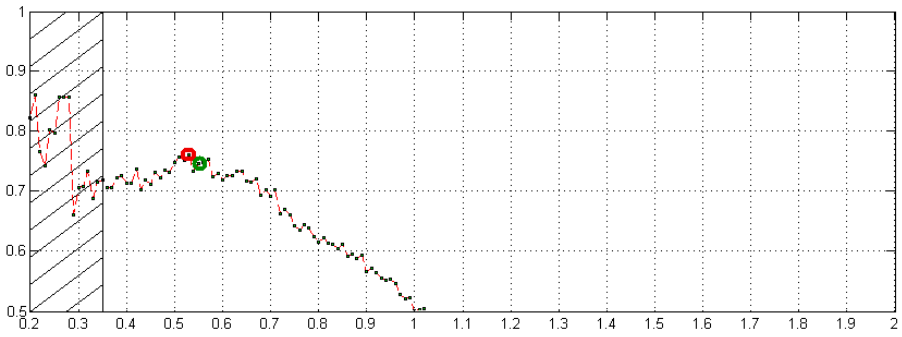


FIG. 5.8 – Résultat de l'analyse d'intervalle — $R = 0.55$

on peut le constater sur la figure 5.8, le fait de ne pas tenir compte des valeurs de r trop petites permet d'éviter d'estimer le taux de redimensionnement à 0.26 ou même à 0.21. Néanmoins, notre définition corrigée pour R n'est plus valide. En effet, nous l'avions redéfini comme étant le premier maximum local trouvé lors du parcours de l'intervalle de droite à gauche (r décroissant). Nous voyons maintenant qu'une solution plus élaborée est nécessaire. Les deux solutions possibles sont les suivantes :

- éliminer la partie de la courbe où les variations de $M(r)$ sont trop importantes ;
- définir une taille de fenêtre de recherche minimum.

Pour remédier à ce problème, la première solution envisageable serait de ne pas tenir compte du début de la courbe et ce, jusqu'à ce que $M(r)$ se stabilise. Cette solution se base sur l'observation suivante : lorsque r est trop petit, des valeurs $M(r)$ consécutives peuvent varier grandement, contrairement au reste de la courbe. Le graphique de la figure 5.9 montre l'évolution de la différence $M(r+1) - M(r)$ pour l'analyse d'intervalle effectuée sur les première ($R = 1.70$) et troisième ($R = 0.56$) images de la séquence. On y voit clairement une stabilisation des courbes au-delà d'un certain seuil.

Une autre solution serait de définir une relation entre la taille de la cible et la borne inférieure de l'intervalle. Par exemple, pour une cible de 50x30 pixels, l'intervalle analysé serait [0.35 ; 2.00] tandis que, si la cible est de 100x60

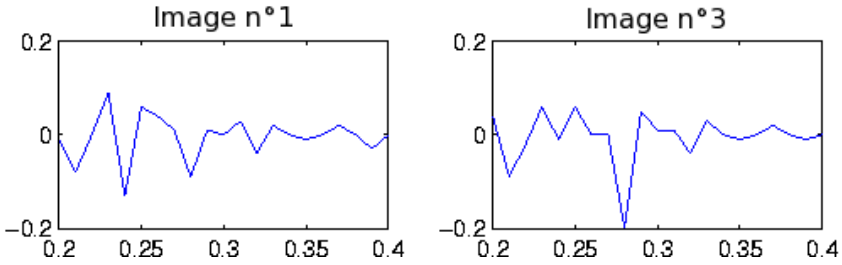


FIG. 5.9 – Graphiques de $M(r+1) - M(r)$

pixels, il serait [0.20 ; 2.00]. Ainsi, nous définirions un nombre minimal de pixels à prendre en compte dans le calcul pour que le CHS fournisse des résultats corrects. Autrement dit, il s'agirait de fixer une taille minimum pour la fenêtre de recherche utilisée lors du balayage d'une image pour la recherche de la cible.

Des expériences plus poussées permettraient de vérifier qu'une des deux solutions expliquées ci-dessus fournit des résultats corrects. Quant à nous, nous ne nous y attarderons pas : la courbe est très bruitée et un lissage ne nous permet pas de résoudre le problème de l'estimation du taux de redimensionnement.

En effet, mis à part le problème des petites tailles, on constate toujours une sous-estimation de la taille correcte : $R_{est} = 0.53$ alors que $R_{correct} = 0.56$. La différence est, dans cet exemple-ci, minime et tout à fait acceptable. Cependant, elle s'est révélée être plus importante dans d'autres tests, tels que celui de la première image de la séquence, affiché sur la figure 5.10 : $R_{est} = 1.57$ alors que $R_{correct} = 1.70$.

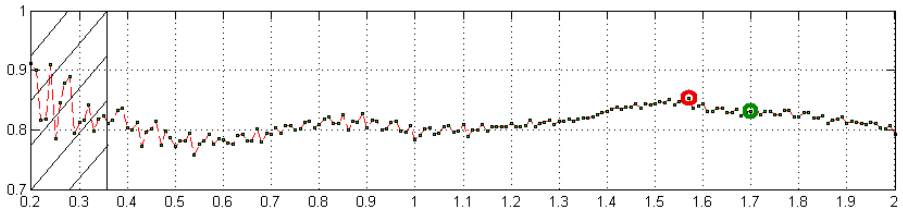


FIG. 5.10 – Résultat de l'analyse d'intervalle — $R = 1.70$

5.3.4 Conclusion

Dans les sections précédentes, nous avons vu que la méthode de l'analyse d'intervalle par itération fournissait des résultats relativement acceptables. L'estimation faite de la taille de la cible dans l'image n'était jamais très éloignée de la valeur correcte.

Cependant, cette technique a également montré ses limites :
- il faut prendre des précautions lors de l'utilisation de taux de redimensionnement peu élevés ;
- lors de l'utilisation d'images réelles, il y a toujours un décalage entre la valeur estimée et la valeur correcte de ce taux.

Pour améliorer ces résultats, il serait peut-être utile d'effectuer un traitement préalable sur l'ensemble des $M(r)$ afin de tenter d'en éliminer le bruit. Il faut donc appliquer un filtre pour lisser la courbe. Cela pourrait en effet permettre d'ajuster certaines valeurs perturbatrices du graphique.

Malgré tout, l'analyse d'intervalle reste une opération beaucoup trop coûteuse en temps de calcul pour que nous puissions envisager de l'employer dans le cadre du temps réel. Ainsi, si on considère un intervalle $[0.2, 2.0]$ avec un pas de 0.1, il est nécessaire d'effectuer le calcul du CHS 18 fois. Or, ce dernier prend 10 à 15 ms (selon la taille de la cible), ce qui nous place bien au-dessus des 40 ms que nous permet le temps réel. On pourrait envisager d'alléger cette technique, mais les résultats qu'elle donne ne justifient pas qu'on s'y attarde plus.

Enfin, cette analyse nous a permis de réaliser qu'il était difficile d'associer le taux R au maximum de similarité du CHS. Dès lors, il est nécessaire de trouver une méthode totalement différente pour estimer la taille de la cible dans la scène.

5.4 « Colour Histogram Footprint »

5.4.1 Description

La méthode du « Colour Histogram Footprint » (CHF dans la suite de ce document) a été présentée dans [8]. Il s'agit d'une méthode d'estimation de la taille sur laquelle nous avons travaillé durant notre stage au TROP. Elle se déroule en 5 étapes :

1. définition d'un taux de redimensionnement r de base,
2. localisation de la cible avec le CHS, en utilisant r,
3. détermination d'une fenêtre de recherche basée sur la position détectée,
4. filtrage de cette fenêtre de recherche afin de créer une empreinte de recherche,
5. analyse de l'empreinte et détermination du nouveau taux r.

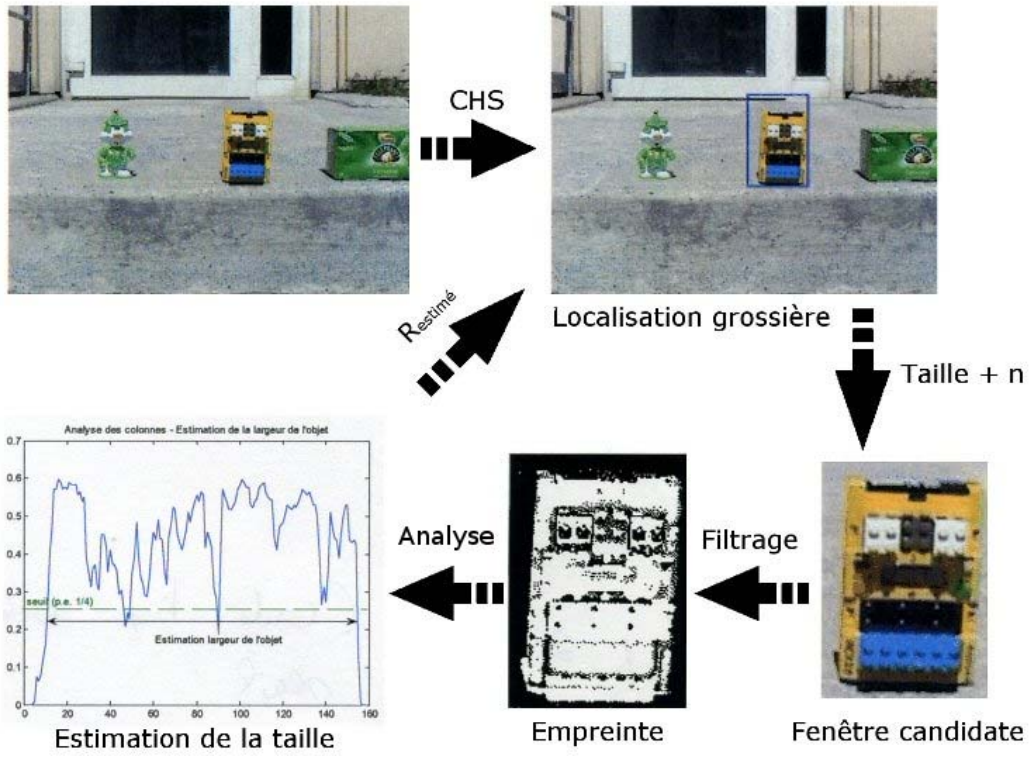


FIG. 5.11 – Fonctionnement du CHF

Le fonctionnement du CHF est schématisé à la figure 5.11. On y voit que la fenêtre de recherche (« fenêtre candidate » sur le schéma) est en fait la cible détectée, mais à une différence près : une bordure de n pixels est ajoutée à la taille de l'image (lors de nos tests, nous avons utilisé $n = 4$). En d'autres termes, si le CHS localise l'objet recherché à la position (x, y) en considérant une taille $L \times H$, alors la fenêtre candidate est située en $(x - n, y - n)$ et ses dimensions sont $(L + 2 \times n) \times (H + 2 \times n)$. La raison de la présence de cette bordure sera expliquée plus loin.

La fenêtre candidate est ensuite analysée afin de produire la fenêtre empreinte. Cette dernière est en fait une image binaire où un pixel q vaut 1 s'il contribue immanquablement à l'intersection d'histogrammes. Un pixel q vaut donc 1 si l'intersection d'histogrammes est modifiée lorsque l'on retire le pixel q de l'ensemble des pixels de l'image :

$$E(q) = \begin{cases} 1 & \text{si } \sum_{i=1}^{n} \min(H_i, M_i) \neq \sum_{i=1}^{n} \min(H_i^{-q}, M_i) \\ 0 & \text{sinon} \end{cases} \tag{5.8}$$

où H^{-q} représente l'histogramme sans le pixel q.

La présence des deux sommes dans cette définition peut faire penser que le calcul de l'empreinte est coûteux en temps de calcul, mais il n'en est rien. En effet, une simplification, démontrée dans [8], donne lieu à la formule suivante, équivalente à la première :

$$E(q) = \begin{cases} 1 & \text{si } H_{i_q} \leq M_{i_q} \\ 0 & \text{sinon} \end{cases} \tag{5.9}$$

où

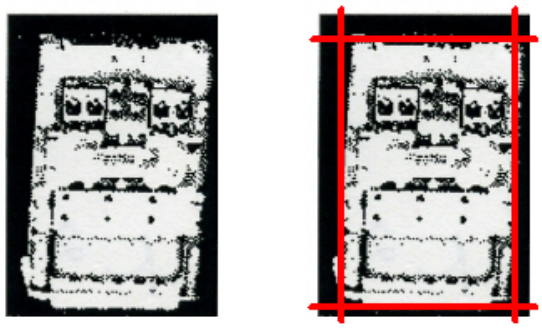
- H et M sont les histogrammes, respectivement, de l'image et la cible.
- i_q est l'index de la couleur du pixel q.

Une fois que la fenêtre empreinte est calculée, il suffit de faire un rapide balayage ligne par ligne puis colonne par colonne pour déterminer la taille de la cible. Le procédé est le suivant : on considère que, lorsqu'un nombre suffisant de pixels valent 1 sur une ligne / colonne, alors il s'agit de l'objet recherché. Dans nos tests, la cible était estimée être comprise entre la première et la dernière ligne / colonne contenant au moins 30% de pixels valant 1. Ce processus est illustré sur la figure 5.12. Le calcul que nous avons alors effectué pour obtenir un taux de redimensionnement r utilisable par l'algorithme du CHS est le suivant :

$$r = \frac{\frac{L_e}{L_c} + \frac{H_e}{H_c}}{2} \tag{5.10}$$

où

- L_e et H_e sont respectivement la largeur et la hauteur calculées sur l'empreinte.
- L_c et H_c sont respectivement la largeur et la hauteur de la cible de référence.



(a) Empreinte (b) Taille estimée

FIG. 5.12 – Détermination de la taille à partir de l'empreinte

Ainsi, l'algorithme du CHF est principalement une opération de filtrage d'une fenêtre candidate produisant une fenêtre empreinte. L'analyse de celle-ci permet d'obtenir une estimation de la taille de l'objet. Effectivement, les pixels « allumés » de l'empreinte (ceux qui valent 1) appartiennent presque exclusivement à la cible recherchée.

Il faut également préciser un point au sujet de la fenêtre candidate. Celle-ci est agrandie de n pixels de chaque côté par rapport à la taille utilisée par le CHS. Le but de cet agrandissement est de permettre à l'algorithme de repérer une augmentation de la taille de la cible. En effet, si on n'effectuait pas cette opération, la taille obtenue serait toujours plus petite ou égale à celle envisagée lors du calcul du CHS. Dès lors, le CHF serait dans l'impossibilité de détecter un accroissement des dimensions de l'objet recherché.

Précisons déjà que les expériences que nous avons faites, et leurs résultats, ont déjà été publiés dans [8]. En effet, une partie des résultats qui y ont été exposés a été obtenue en utilisant l'application C++ sur laquelle nous avons travaillé durant notre stage. Globalement, on peut dire que les résultats de la méthode sont satisfaisants.

Dans les deux sections suivantes, nous allons étudier les résultats de l'algorithme du CHF sur deux exemples concrets. Nous l'envisagerons dans le cas de séquences d'images : le taux de redimensionnement r calculé sur une image sera réutilisé pour le calcul du CHF sur l'image suivante. Ainsi, lorsque le CHF fait appel au CHS pour la $i^{ème}$ image, il lui passera comme paramètre

r_{i-1} pour le taux de redimensionnement. Nous considérerons que r_0 vaut 1.

Les deux tests que nous allons faire ci-après ont chacun un but différent qui est :

1. montrer l'avantage qu'apporte la méthode du CHF au CHS,
2. montrer le problème principal de la méthode du CHF.

5.4.2 Test n°1

Comparons le déroulement du CHS et du CHF sur une séquence vidéo. Nous nous attarderons plus précisément sur les deux images de la figure 5.13. La cible à localiser dans les images est l'étiquette de cette bouteille,

(a) Image n°1 ($100^{ème}$ dans la vidéo) (b) Image n°2 ($108^{ème}$ dans la vidéo)

FIG. 5.13 – Séquence de deux images

l'image de référence ayant été tirée de la première image de la vidéo, où la bouteille était bien plus près (toutes les images analysées et les résultats sont disponibles en annexe B).

Malgré la différence de taille assez importante par rapport à la cible de référence, l'algorithme du CHS parvient à localiser correctement l'étiquette de la bouteille de limonade sur l'image n°1. Cependant, le CHF, grâce à l'estimation de R, donne des résultats encore plus précis. Nous n'afficherons pas ici les positions obtenues, elles sont disponibles en annexe B. Intéressons-nous plutôt à la deuxième image de la séquence, sur laquelle la bouteille est encore plus éloignée. Les résultats obtenus sont affichés sur la figure 5.14.

Comme on le voit, le CHF localise correctement l'étiquette de la bouteille de limonade tandis que le CHS échoue. L'erreur est compréhensible puisque

(a) CHS (b) CHF

FIG. 5.14 – Résultats du CHS et du CHF sur l'image n°2

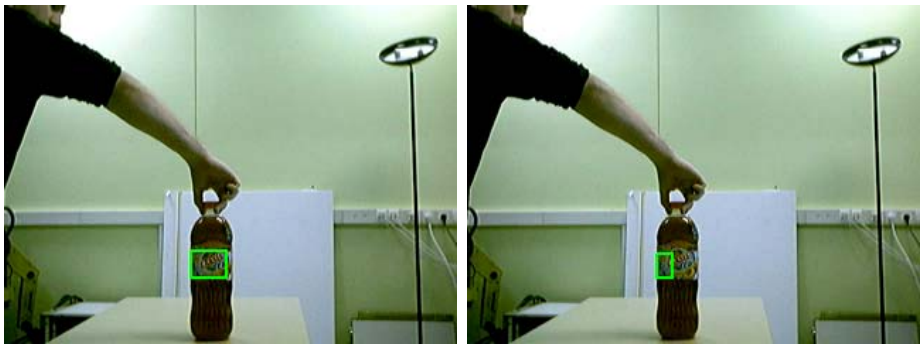
les couleurs du dictionnaire sont semblables à celles de la cible. Dans le cas de l'image précédente, la bouteille était encore assez proche et les quantités de chaque couleur correspondaient plus à la référence que pour le dictionnaire. Cependant, un trop grand éloignement a eu pour effet de favoriser ce dernier sur l'image n°2.

Le CHF réussit sans aucun problème puisqu'il utilise le taux de redimensionnement calculé sur l'image précédente dans la séquence. Ainsi tout au long de la séquence, la cible est localisée avec une précision acceptable. Les dimensions estimées sont également correctes, à quelques pixels près.

Cet exemple nous a donc permis de mettre en avant l'importance de l'utilisation d'un algorithme d'estimation de la taille. Nous avons pu constater que le CHF est une solution possible qui nous donne des résultats plutôt satisfaisants. De plus, la simplicité du calcul de l'empreinte nous permet de rester dans les limites du temps réel (20 ms en moyenne, soit une augmentation de 5 ms par rapport au CHS).

5.4.3 Test n°2

Dans cet exemple, nous allons montrer le problème majeur qui survient lors de l'utilisation du CHF. Pour cela, considérons les deux images de la figure 5.15. La première correspond à la $4^{ième}$ image d'une séquence vidéo tandis que la seconde en est la $6^{ième}$. Deux autres images de cette vidéo sont disponibles également en annexe B. Dans cette séquence, une bouteille de limonade est progressivement éloignée.



(a) Image n°1 (b) Image n°2

FIG. 5.15 – Images posant problème pour le CHF

On remarque sur l'image n°1 que la cible — en l'occurrence, l'étiquette de la bouteille de limonade — n'est pas entièrement encadrée par le rectangle. Autrement dit, l'algorithme du CHF a légèrement sous-estimé la taille de la cible. Cependant, le problème est amplifié sur la deuxième image : l'estimation échoue lamentablement.

Effectivement, la taille calculée pour la première image est incorrecte, mais elle est réutilisée pour l'image suivante dans la séquence. L'algorithme part donc sur de mauvaises bases pour celle-ci et la situation ne peut qu'empirer. La seule possibilité d'amélioration tient du fait qu'on considère une fenêtre candidate plus grande que la dernière taille estimée. Malheureusement, cela permet de gérer un accroissement des dimensions de l'objet uniquement lorsque cela se fait progressivement. Dès lors, une correction automatique de l'erreur aurait pu être faite entre la première image et la suivante, mais la sous-estimation sur l'image n°2 est bien trop grave.

On peut cependant se poser les questions suivantes : l'erreur aurait-elle pu se corriger d'elle-même dès l'image n°1 et pourquoi ne l'a-t-elle pas fait ? L'explication est simple. Pour le comprendre, il suffit de se référer à la figure 5.16 qui représente les empreintes calculées pour les deux images ci-dessus. On constate dès que les parties du bas et de gauche seront négligées lors de l'analyse de l'empreinte. De plus, les images donnent à penser que la cible se prolonge en haut à droite, ce qui n'est bien sûr pas pris en compte par l'algorithme.

(a) Empreinte de l'image n°1 (b) Empreinte de l'image n°2

FIG. 5.16 – Empreintes calculées pour les deux images de la figure 5.15

Une amélioration possible du CHF serait de repérer lorsqu'un cas pareil se produit et de le gérer en effectuant, par exemple, un recentrage sur la cible avant d'en estimer la taille.

5.4.4 Conclusion

Dans les sections précédentes, nous avons vu que l'algorithme du « Colour Histogram Footprint » donne d'assez bons résultats. Effectivement, il permet d'estimer la taille de la cible avec une précision très satisfaisante. De plus, nous avons vu que, lorsqu'on détermine correctement les dimensions de l'objet recherché, la localisation était beaucoup plus performante.

En outre, cette technique nécessite peu de temps de calcul. Puisqu'elle repose sur l'utilisation du CHS, elle ne saurait être plus rapide que ce dernier. Ainsi, le CHS nécessite en moyenne 15 ms tandis que le CHF en prend 20 ms. Cela reste donc toujours dans le cadre du temps réel. Cependant, l'un comme l'autre sont dépendants de la taille de la cible et de l'estimation qui en est faite. Dès lors, pendant nos tests, nous avons constaté des temps atteignant parfois les 50 ms, ce qui est à la limite de l'acceptable. Il s'agissait cependant de rares pointes et, en général, il était possible de suivre le rythme de la caméra. Pour rappel, cette dernière produit 25 images par seconde, ce qui fait un maximum de 40 ms de traitement autorisé par image.

Malgré ces bonnes performances, nous avons constaté un grave problème dans cette technique : une trop importante sous-estimation de la taille de la cible a tendance à éloigner définitivement le CHF de la bonne solution. Ainsi, lorsque ce phénomène se produit, il y a peu de chances que la taille

correcte soit trouvée dans les images suivantes de la séquence.

Un autre problème est dû à l'interprétation du contenu empreinte. Considérons par exemple la figure 5.17. On peut déduire la taille de la cible (encore

FIG. 5.17 – Mauvaise interprétation du contenu de l'empreinte

l'étiquette d'une bouteille de limonade) d'un simple coup d'oeil. Cependant, la méthode que nous utilisons risque de déterminer une taille trop grande à cause de la ligne du dessus qui contient un grand nombre de pixels valant 1. Cela dépend du seuil que l'on définit (dans nos tests, nous considérions qu'il fallait 30% de pixels blancs).

Heureusement, ces problèmes ne sont sans doute pas insolubles et on peut raisonnablement espérer y trouver une solution. La méthode du CHF est en tout cas très prometteuse pour la détection et le suivi d'une cible en temps réel.

5.5 Conclusion

Dans le cadre du suivi d'un objet dans une séquence d'images, il est crucial de pouvoir estimer la taille de la cible. En effet, tout au long de ses mouvements, celle-ci peut se rapprocher ou s'éloigner de la caméra. Dès lors, les algorithmes de localisation, qui se fondent sur une référence établie au début de la vidéo, partent sur de mauvaises bases. Ainsi, si les dimensions de l'objet recherché diminuent de moitié, il ne pourra pas être détecté par les techniques habituelles.

Dans l'optique de pallier ce problème, la méthode du CHS a été améliorée : un paramètre permettant de définir un taux de redimensionnement y a été ajouté. Grâce à ce paramètre, il est maintenant possible de détecter une

cible dont la taille a changé par rapport à l'image de référence. Pour ce faire, nous partons du principe que, si les dimensions d'un objet diminuent (resp. augmentent), alors le contenu de chaque « bin » de l'histogramme varie dans les mêmes proportions.

Cependant, la prise en compte du redimensionnement de l'objet recherché par le CHS ne suffit pas à résoudre le problème. Effectivement, il faut désormais calculer la valeur de ce paramètre. Dans cette optique, nous avons étudié en détail deux méthodes :

- l'analyse d'intervalle,
- le « Colour Histogram Footprint ».

La première méthode employait le CHS en lui fournissant plusieurs valeurs (comprises dans un intervalle spécifique) pour le taux de redimensionnement. Comme nous l'avons vu, les résultats ne sont pas fondamentalement mauvais : les dimensions calculées sont proches de la réalité, mis à part une sous-estimation systématique qui pourrait sans doute être corrigée. Cependant, l'utilisation intensive du CHS (18 fois pour l'intervalle [0.2 ; 2.0] avec un pas de 0.1) rend cette méthode trop coûteuse en temps de calcul. Son étude nous a néanmoins permis de saisir tous les tenants et les aboutissants du problème de l'estimation des dimensions de la cible dans l'image.

Ensuite, nous avons abordé la méthode du « Colour Histogram Footprint » sur laquelle nous avons également travaillé durant notre stage. Cette technique se base sur une localisation grossière de la cible pour définir une zone dont on va calculer l' « empreinte ». Cette dernière opération consiste en un filtrage qui permet de savoir quels pixels ont contribué à coup sûr à l'intersection d'histogrammes. Une fois l'empreinte calculée, on peut l'analyser pour déterminer les dimensions de la cible. Nous avons vu que cette méthode fournissait d'assez bons résultats tout en respectant les contraintes du temps réel. Elle présente certains inconvénients auxquels il faut encore remédier. Ainsi, une interprétation différente de l'empreinte pourrait sans doute mener à de meilleurs résultats.

En conclusion, cette partie nous a permis de voir qu'il ne fallait pas négliger l'estimation de la taille de la cible dans l'image. Nous avons pu analyser ce problème et y fournir une solution, le CHF. Bien qu'imparfaite, celle-ci donne lieu à de bonnes estimations, proches de la réalité. Des recherches plus poussées autour de la méthode du CHF pourraient permettre de l'améliorer.

Chapitre 6

Indicateurs

6.1 Introduction

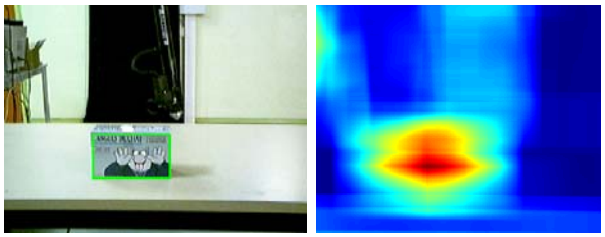
Étant donné la nature du résultat final de l'algorithme du CHS (maximum de la surface de similarité), une solution est toujours trouvée. La solution représente la position la plus probable de la cible selon la surface de similarité. Il est possible que l'algorithme se trompe et ne trouve pas correctement l'objet recherché mais un objet parasite. Cette erreur peut se produire lorsque la scène de l'image rend difficile son analyse (illumination, taille de l'objet par rapport à sa référence, ...) ou lorsque la cible n'est simplement pas dans l'image. Dans les deux cas, il est nécessaire de pouvoir se rendre compte de cette erreur. Il vaut mieux savoir que l'on ne trouve pas la cible (et peut-être prendre des mesures afin de la retrouver) que de suivre une pseudo-cible. Nous devons donc quantifier la qualité des solutions afin de pouvoir identifier de la façon la plus efficace les solutions erronées.

Plusieurs indicateurs ont été testés afin de trouver une valeur ou un ensemble de valeurs qui permettent d'accepter ou de rejeter une solution. Chaque indicateur sera jugé sur sa capacité à définir une limite pour séparer au mieux les bonnes des mauvaises solutions ainsi que sur la complexité du calcul nécessaire à sa réalisation. Comme nous le verrons plus loin, la paramétrisation de cette limite est assez problématique. Chacun indicateur sera évalué individuellement puis nous étudierons la possibilité de développer une solution qui couple plusieurs indicateurs.

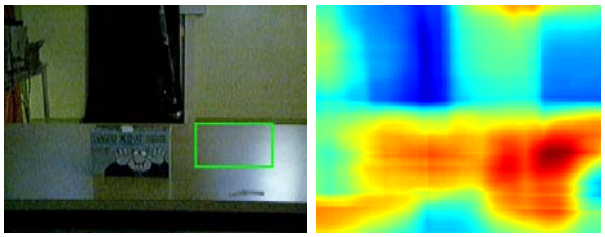
6.2 Maximum de la surface de similarité

Utiliser le maximum de la surface de similarité comme indicateur est simple car directement accessible à la fin du CHS. Ce maximum, qui varie entre 0 et 1, représente la « ressemblance » entre la cible de référence et la solution dans l'image. Nous pourrions donc établir que, si le maximum n'est pas plus grand qu'une certaine limite x, la solution est mauvaise puisqu'elle ne ressemble pas assez à la cible.

Malheureusement, comme nous l'avons vu précédemment, toute modification de la lumière ou de la taille de la cible fait fortement diminuer la valeur du CHS. Cela entraîne un nombre important de faux négatifs dès que la limite x devient un peu trop haute. A l'opposé, si la limite x n'est pas assez haute, de nombreux faux positifs peuvent passer le test. En effet, il suffit d'une zone de l'image qui ait la couleur majoritaire de la cible (si elle en a une) pour avoir une valeur de similarité relativement haute. Nous pouvons voir sur un exemple (figure 6.1) que deux valeurs de maximum presque équivalentes correspondent à deux situations opposées.



(a) Résultat correct (b) Maximum 0.5



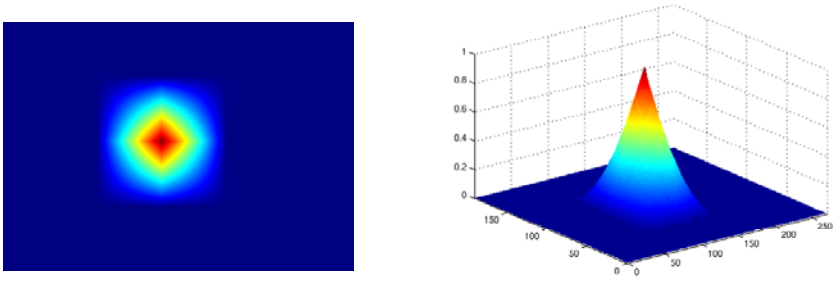
(c) Résultat incorrect (d) Maximum 0.48

FIG. 6.1 – Exemple du maximum comme indicateur

Suivant ces conclusions (intuitives et testées sur de nombreuses images), le maximum de la surface de similarité ne fournit pas un critère suffisant pour séparer les solutions correctes des solutions erronées.

6.3 Qualité du pic

La surface de similarité idéale donnerait une valeur de 1 à l'endroit où se trouve l'objet et une valeur proportionnelle au nombre de pixels qui se trouvent dans la fenêtre de recherche et qui appartiennent à la cible (figure 6.2). Nous aurions une surface plane avec un pic dont le sommet serait la solution. Nous pourrions donc essayer d'évaluer si la surface de similarité que nous avons est proche de cet idéal ou pas.



(a) Surface 2D (b) Surface 3D

FIG. 6.2 – Exemple de surface idéale

A cette fin, nous utiliserons un indicateur facilement calculable qui, globalement, représente la qualité du pic.

$$qpic = maximum(SurfSim) - moyenne(SurfSim) \tag{6.1}$$

Plus *qpic* est proche de zéro, plus l'ensemble de la surface est proche du maximum et le pic est faiblement prononcé. Dans le cas contraire, nous aurons un beau pic bien prononcé par rapport au reste de la surface. Le travail à réaliser afin d'avoir *qpic* est très faible : le maximum est connu et la moyenne peut être calculée au fur et à mesure du CHS.

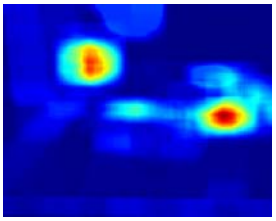
Cet indicateur fonctionne beaucoup mieux que le maximum seul. Dans l'exemple de la figure 6.1, le premier cas donne un *qpic* de 0,4 alors que le

second donne une valeur de *qpic* de 0.22. Même s'il permet de réaliser une classification de meilleure qualité, il garde certaines faiblesses dérivées de l'indicateur du maximum. En effet, lorsqu'une diminution du maximum est due à un changement de l'environnement, la valeur de *qpic* chute et peut descendre en dessous de la limite fixée même si la cible est toujours trouvée correctement. Nous pourrions combler ce défaut en normalisant *qpic* par $maximum(SurfSim)$. *qpic* serait ainsi moins sensible aux chutes globales de la surface mais deviendrait complètement inutile lorsque l'ensemble de la surface est très basse. Nous pourrions, par exemple, avoir une surface dont le $maximum(SurfSim) = 0.01$ et la $moyenne(SurfSim) = 0.001$ qui donnerait un *qpic* de 0.9.

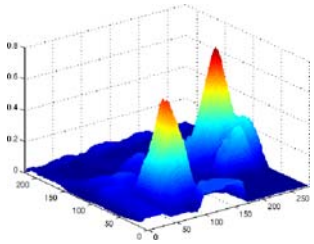
Nous ajouterons à ce problème quelques cas où *qpic* a une valeur élevée mais ne représente pas une surface où un seul pic marqué est présent. Par exemple, lorsque deux pics bien marqués sont présents, la valeur de *qpic* sera bonne alors que la solution n'est pas excellente (lequel des deux pics est la bonne solution ?). Nous pouvons observer ce phénomène sur la figure 6.3. Deux pics sont clairement présents sur la surface de similarité. Le *qpic* vaut

(a) Image originale

(b) Cible recherchée



(c) Surface de similarité



(d) Surface 3D

FIG. 6.3 – Exemple de plusieurs pics

0.67 et le *qpic* normalisé 0.89 ce qui donne à penser que nous avons une excellente solution. Bien que nous ayons effectivement une solution correcte, nous avons aussi un second pic qui aurait pu tout aussi bien être la solution recherchée. Le cas où le parasite est choisi à la place de l'objet désiré arrive très rapidement et *qpic* ne permet pas de le déceler. *qpic* permet donc de savoir si la solution donnée est au moins un pic mais pas de savoir si c'est le bon.

6.4 Qualité de l'empreinte

L'empreinte (définie au chapitre 5), même si elle n'a pas été conçue dans ce but, pourrait être utilisée comme indicateur. Elle représente les pixels dont on est pratiquement sûr qu'ils appartiennent à la cible. Plus elle a de pixels « allumés », plus notre solution devrait avoir de chances d'être la bonne. La qualité de l'empreinte est donc simplement donnée par la moyenne des pixels « allumés » de l'empreinte.

$$qemp = \frac{\sum_{i=1}^{largeur\_cible} \sum_{j=1}^{longeur\_cible} empreinte_{ij}}{largeur\_cible * longeur\_cible} \tag{6.2}$$

Le temps de calcul est plus lourd que les indicateurs précédents mais reste acceptable puisqu'il suffit de calculer une seule empreinte pour la solution trouvée. De plus, si le CHS est déjà complété par l'algorithme de recherche de taille, le temps de calcul supplémentaire est alors nul.

Après quelques jeux de tests, nous pouvons voir que cet indicateur est fortement corrélé avec $maximum(SurfSim)$ et apporte très peu d'informations supplémentaires. Dans le jeu de tests B (voir Annexe D), on peut remarquer un comportement équivalent de $maximum(SurfSim)$ et *qemp* (figure 6.4) excepté pour les images n°5 et n°12 (figure 6.5). La scène de l'image 5, même si elle est exagérément mal éclairée, démontre un des rares cas où *qemp* peut faire la différence. La lumière crée une zone où les pixels ont la même couleur que la couleur majoritaire de la cible (le gris). Cette zone réussit à décrocher un très bon score de CHS grâce à cela mais *qemp* chute car cette couleur est trop présente et il n'y en a quasiment aucune autre de la cible. L'image n°12 réagit de la même manière, de façon plus

mesurée, car de nombreux pixels du fond interfèrent avec l'intersection et ne sont pas pris en compte par l'empreinte.

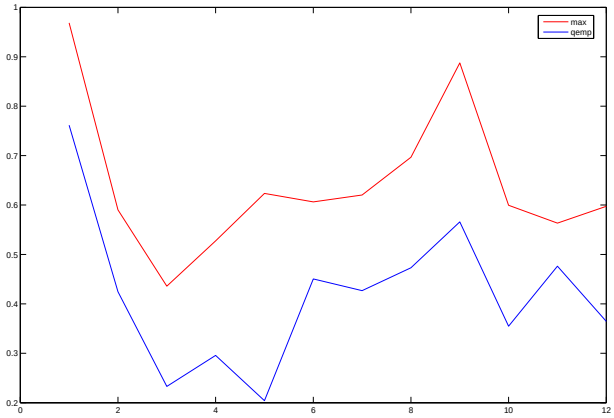
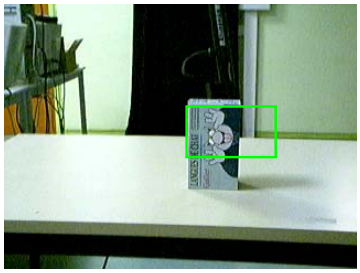


FIG. 6.4 – Comparaison du maximum et de la qualité de l'empreinte

(a) Image n°5



(b) Image n°12

FIG. 6.5 – Images donnant un maximum et une qualité d'empreinte différents

Nous pouvons donc considérer l'intérêt de *qemp* relativement limité et ne l'utiliser que lorsque très peu d'efforts sont nécessaires (lors de l'utilisation du CHF par exemple).

6.5 Corrélogramme

Le corrélogramme, développé dans la section 2.5.3, permet de prendre en compte en plus de la composition des couleurs, les positions des pixels de couleur entre eux. Cette méthode est plus lente que le CHS mais pourrait être utilisée de façon localisée sur la solution proposée par le CHS pour en déterminer plus finement sa qualité.

Si théoriquement elle devrait pouvoir départager plus facilement les mauvaises solutions des bonnes, en pratique les résultats du corrélogramme sont difficilement exploitables. Comme nous avons pu le remarquer sur la figure 2.17 de la section sur le corrélogramme, l'intersection de corrélogrammes est faiblement discriminante (énormément de rouge sur la figure). Le seul cas où nous pouvons remarquer une réaction forte du corrélogramme est constaté lorsque la solution est seulement composée d'une ou deux couleurs de la cible. Ce cas, qui donne une grande valeur de CHS, peut être déjà écarté grâce à la qualité de l'empreinte. Le test du corrélogramme ne permet donc pas de justifier son coût de calcul supplémentaire.

6.6 Indicateur d'indicateurs

Toutes les valeurs testées ci-dessus sont relativement sensibles aux changements d'illumination de la scène. Il serait donc utile de pouvoir définir un certain niveau d'illumination nécessaire afin de pouvoir garantir un minimum le résultat des indicateurs. La luminosité de la scène peut être facilement calculée en effectuant la moyenne de la luminosité de chaque pixel.

$$Lum = \sum_{i=1}^{num\_pixels} V_i \tag{6.3}$$

où $V_i = \max(R_i, G_i, B_i)$

Deux utilisations sont possibles de Lum, soit de façon absolue, soit de façon relative à l'illumination de la cible de référence. Dans le premier cas, nous définissons un minimum d'illumination que la scène doit avoir pour pouvoir avoir un résultat satisfaisant du CHS et de ses indicateurs. Dans le deuxième cas, nous définissons un maximum de différence entre Lum et Lum_{ref}. En définissant cette différence par $dlum = Lum - Lum_{ref}$, la borne sera négative. En effet, lorsque $dlum$ est positif la scène est mieux éclairée

que la référence. Ce cas est en général peu problématique puisque les couleurs de la scène restent bien différenciées. Par contre, lorsque *dlum* est négatif, la scène est moins bien éclairée et peut commencer à poser problème.

Les deux approches peuvent être conjointement utilisées afin de cerner un cadre minimal d'utilisation de l'algorithme du CHS et ainsi garantir une validité minimale des indicateurs utilisés.

6.7 Utilisation conjointe des indicateurs

Si aucun indicateur présenté ci-dessus ne permet, isolé, de déterminer efficacement si une solution du CHS est bonne ou mauvaise, une combinaison des différents indicateurs devrait pouvoir étoffer la capacité de décision. Nous pouvons, par exemple, combiner la *qpic* normalisée et $maximum(SurfSim)$ afin de définir un jeu de valeurs qui délimite plus finement les solutions. Nous pouvons aussi ajouter un test sur la *qemp* si le CHF est calculé afin d'éliminer quelques cas récalcitrants supplémentaires. Le tout serait conditionné par un test sur la luminosité afin de savoir s'il est nécessaire d'aller plus loin.

Ce type de combinaison permet de voir si :
- on est capable de détecter quelque chose (test de luminosité),
- la solution ressemble suffisamment à la cible de référence (test du maximum),
- la solution ressort par rapport à son environnement (test de qualité du pic),
- la solution n'est pas une zone de couleur majoritaire de la cible (test de qualité de l'empreinte).

Nous pouvons raffiner cette combinaison en effectuant un traitement conditionnel des différentes limites fixées aux indicateurs. Par exemple, nous pourrions être plus exigeants sur les valeurs des indicateurs lorsque la lumière de la scène est au moins aussi bonne que celle de la cible référence. En effet, un maximum de 0.5 serait plus impressionnant pour une scène faiblement éclairée.

6.8 Paramétrage pratique

Après avoir testé intensivement les différents indicateurs présentés ci-dessus, nous sommes arrivés à ce modèle de test pour une quantification RGB de 333 :
Soient,
lum, la luminosité de la scène
lum_{ref}, la luminosité de la cible de référence
$dlum$, la différence entre lum et lum_{ref}
max, le maximum de la surface de similarité
$qpic$, la qualité du pic
Alors

$Si \quad lum < 60$
résultat = invalide

Si $dlum > -(lum_{ref})/4$
Si $max > 0.4$ et $qpic > 0.5$
résultat = valide
Sinon
résultat = invalide
Sinon si $dlum > -(lum_{ref})/2$
Si $max > 0.3$ et $qpic > 0.4$
résultat = valide
Sinon
résultat = invalide
Sinon
résultat = invalide

Ce pseudo-code utilise tous les indicateurs aisément calculables et les utilise de façon dissociée. Nous fixons des limites plus fortes lorsque la scène est mieux ou un peu moins bien éclairée ($dlum > -(lum_{ref})/4$) et des limites un peu plus faibles lorsque la différence commence à se faire sentir ($-(lum_{ref})/2 < dlum < -(lum_{ref})/4$). Si la scène est mal éclairée ou trop

Image	max	qpic	lum	dlum	résultat
Image A.1	0.8955	0.7635	140	65	valide
Image A.2	0.3514	0.6540	51	-23	invalide
Image A.3	0.7382	0.7135	131	57	valide
Image A.4	0.8480	0.7682	152	77	valide
Image A.5	0.8495	0.7775	151	77	valide
Image A.6	0.6935	0.4501	131	57	invalide
Image A.7	03541	0.4258	41	-32	invalide
Image A.8	0.6944	0.4499	118	44	invalide
Image A.9	0.4945	0.4709	161	87	invalide
Image A.10	0.5079	0.4557	157	83	invalide
Image B.1	0.9525	0.7634	105	21	valide
Image B.2	0.5241	0.7276	143	60	valide
Image B.3	0.4117	0.7061	142	59	valide
Image B.4	0.3730	0.5118	104	20	invalide
Image B.5	0.6154	0.5732	29	-55	invalide
Image B.6	0.5078	0.5243	41	-42	invalide
Image B.7	0.4622	0.4189	59	-24	invalide
Image B.8	0.6015	0.7127	114	30	valide
Image B.9	0.8452	0.732	118	35	valide
Image B.10	0.5526	0.695	124	41	valide
Image B.11	0.4973	0.5945	119	36	valide
Image B.12	0.5352	0.6271	127	43	valide

TAB. 6.1 – Résultats des jeux de tests

différenciée par rapport à la cible($dlum < -(lum_{ref})/2$ ou $lum < 60$), nous considérons cette image comme intraitable.

Ce modèle marche très bien pour la grande majorité des images que nous avons essayées. Les séries de tests A et B (voir annexe C et D), par exemple, donnent des résultats satisfaisants (voir tableau 6.1).

Dans chacun des cas, le résultat du CHS est correctement évalué. Seul le test sur l'image A.6 évalue la solution du CHS comme mauvaise alors que la cible est correctement identifiée. Ceci est dû à un objet parasite qui fait fortement chuter la qualité du pic. Ce type de faux négatif est difficile à contourner mais si la solution est bien la bonne, ce n'est peut-être qu'un coup

de chance. Nous pouvons donc voir que la détection de mauvaise solution fonctionne relativement bien.

Il est cependant à noter deux problèmes. Premièrement, les valeurs utilisées sont fixées pour une certaine quantification. Changer la quantification entraînerait de forts changements dans les valeurs des indicateurs qui rendraient ce modèle caduque. Deuxièmement, en temps réel, les images récupérées par une « webcam » varient énormément sans aucune interaction extérieure (aucune modification de la scène ni de la position de la caméra). Cela peut entraîner une variation importante des indicateurs qui risquent d'osciller entre les limites. Nous pouvons donc avoir, dans un cas extrême, une image acceptée puis rejetée en alternance. Ce type de comportement reste assez délicat lorsque la caméra doit exécuter un suivi de cible en temps réel.

6.9 Conclusion

Comme le CHS trouve automatiquement une solution, il est possible que la zone trouvée dans l'image ne soit absolument pas la cible recherchée. Lors de notre stage, nous avons cherché, à partir des informations disponibles (surfaces de similarité, image analysée, résultats du CHS, ...), une possibilité d'identifier les solutions erronées résultantes du CHS. Plusieurs indicateurs ont été testés individuellement afin d'essayer d'en extraire le maximum d'informations. Aucun de ces indicateurs n'a pu, seul, résoudre le problème.

Nous avons donc trouvé et testé, à partir de tous ces indicateurs simples, un moyen de différencier les bonnes solutions des mauvaises. Cette méthode doit, cependant, être adaptée à chaque quantification particulière. Malgré ces quelques limitations, elle permet d'avoir une bonne estimation de la valeur de la solution.

Conclusion

Nous avons présenté dans ce mémoire différentes techniques de localisation d'une cible de référence dans une image couleur. Nous avons notamment détaillé l'algorithme du CHS sur lequel le groupe TROP travaille actuellement. Cet algorithme présente quelques faiblesses que nous nous sommes attachés à corriger.

Dans un premier temps, nous avons situé notre travail par rapport au laboratoire TROP et à ses objectifs. Pour rappel, le problème de l'asservissement visuel d'un robot y est étudié. Dans ce contexte, nous avons travaillé à la validation d'un algorithme de localisation. En effet, certaines lacunes doivent être éliminées avant de pouvoir passer à la phase d'intégration dans une plate-forme robotique.

Dans ce but, lors de notre stage, nous avons travaillé au développement d'une application permettant de tester, en temps réel, divers algorithmes sur des séquences vidéo. Nous avons doté ce programme de toutes les fonctionnalités nécessaires afin de pouvoir comparer le comportement des différentes techniques de localisation présentées dans ce document.

Dans le chapitre 2, nous avons exposé les notions essentielles à la bonne compréhension du problème de la localisation couleur. Une présentation de différentes techniques y a également été effectuée.

La suite de notre mémoire a été consacré à l'étude approfondie des lacunes de l'algorithme du CHS. Les chapitres 3 à 6 se sont focalisés sur ces faiblesses et des solutions y ont été proposées.

Ainsi, nous avons exploré les différents modes de représentation de la couleur. Ce travail de recherche nous a permis de comparer les résultats du CHS sur un certain nombre d'espaces couleur. De cette comparaison, nous avons pu conclure que les espaces RGB et HSV étaient les meilleurs choix

pour l'utilisation du CHS.

Ensuite, le problème de la robustesse à l'illumination, déjà cité dans le chapitre précédent, a été étudié en détails. Nous avons souligné l'importance d'un bon choix d'espace couleur et sa quantification ainsi que le besoin de complément à l'algorithme de base. Deux possibilités ont été étudiées : étendre l'intersection à un voisinnage, grâce à l'EMD et au CMAC, ou créer un historique de références. Ces deux méthodes ne sont pas totalement concluantes, mais elles méritent d'être approfondies, car elles constituent des pistes intéressantes pour trouver des solutions à ce problème.

Le chapitre suivant a été consacré à l'estimation de la taille de la cible. Nous y avons vu que, dans une séquence vidéo, il était nécessaire de pouvoir estimer assez précisément les dimensions de la cible dans l'image. Nous avons étudié les méthodes d'analyse d'intervalle et du CHF. La première nous a surtout permis de mieux cerner le problème, sans vraiment apporter de solutions. Le CHF constitue une approche intéressante, mais les recherches doivent continuer afin d'améliorer ses résultats.

Enfin, nous avons étudié le problème de l'évaluation des résultats du CHS. A partir d'indicateurs facilement extraits des données, nous avons pu établir des critères pour vérifier la validité des résultats. Ces critères sont cependant dépendants de la quantification et de l'espace couleur, il est donc nécessaire de les adapter pour chaque modification de ces paramètres.

Nous avons donc présenté des solutions isolées pour combler certaines failles de l'algorithme du CHS. D'autres problèmes sont toujours présents, mais un travail futur pourrait, tout en explorant les pistes proposées, étudier la mise en commun de ces solutions afin d'analyser le comportement de ce CHS amélioré.

En conclusion, notre étude a permis de dégager des solutions, ou du moins des pistes de solutions, pour les difficultés rencontrées lors de la localisation basée sur des informations couleur. Ces problèmes ont été abordés de manière originale par rapport à l'existant, car peu de scientifiques mettent actuellement l'accent sur une contrainte de temps réel. De plus, les approches envisagées ne faisaient pas d'hypothèses simplificatrices sur l'illumination de la scène, la taille de l'objet, le décor, ...

Bibliographie

[1] J.S. Albus. A new approach to manipulator control : The cerebellar model articulation controller (CMAC). *Transactions of the ASME*, pages 220–227, September 1975.

[2] Julie Anciaux. Gestion électronique de documents multimédia : Techniques de recherche et d'indexation. Mémoire, Facultés Universitaires Notre-Dame de la Paix, Namur (Belgique), 2004.

[3] O. Bernier, M. Collobert, et D. Collobert. Détection et suivi robuste de visages en temps réel. In *Compression et Représentation des signaux audiovisuels*, Lyon (France), 2003.

[4] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 232–237, Santa Barbara, California (USA), 1998.

[5] Thomas Boutell. *RFC 2083 - Portable Network Graphic Specification Version 1.0*, 1997.

[6] Michael Boyle. The effects of capture conditions on the camshift face tracker. Technical report, University of Calgary, Calgary (Canada), 2001.

[7] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1998.

[8] J.L. Buessler, J.P. Urban, G. Hermann, and H. Kihl. Color histogram algorithms for visual robot control. *International Journal on Robotics and Automation*, 2004.

[9] J.L. Buessler, J.P. Urban, G. Hermann, and H. Kihl. Color histogram similarity for robot-arm guiding. In *International Conference on Complex*

Systems Intelligence and Modern Technological Applications, Cherbourg (France), 2004.

[10] Peng Chang and John Krumm. Object recognition with color cooccurrence histograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, Fort Collins, CO (USA), 1999.

[11] Scott Cohen. *Finding Color and Shape Patterns in Images*. PhD thesis, Stanford University, Stanford (USA), 1999.

[12] Geert De Cubber, Hichem Sahli, Hong Ping, and Eric Colon. A color constancy approach for illumination invariant color target tracking. In *IARP Workshop on Robots for Humanitarian Demining*, Vienne (Autriche), 2002.

[13] Adrian Ford and Alan Roberts. Colour space conversions, 1998. `http://www.poynton.com/PDFs/coloureq.pdf` (Dernière visite : 13 mai 2005).

[14] G. Hermann, D. Greboval, H. Kihl, and J.P. Urban. Evaluation of color-based techniques for robotic positioning tasks. In *The 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, Florida (USA), 2004.

[15] Jing Huang. *Color-Spatial Image Indexing and Applications*. PhD thesis, Cornell University, 1998.

[16] Robyn Owens. Vision course, 1997. `http://www.csse.uwa.edu.au/~robyn/Visioncourse/colour/lecture/colour.ht%ml` (Dernière visite : 13 mai 2005).

[17] P.C. Parks and J. Militzer. Improved allocations of weights for associative memory storage in learning control systems. In *First IFAC Symposium on Design Methods for Control Systems*, volume 2, pages 777–782, Zürich, September 1991. Pergamon Press.

[18] Greg Pass and Ramin Zabih. Comparing images using joint histograms. *Multimedia Systems*, 7(3) :234–240, 1999.

[19] Zoran Pecenovic, Minh Do, Serge Ayer, and Martin Vetterli. New methods for image retrieval. In *ICPS'98 Congress on Exploring New Tracks in Imaging*, pages 242–246, Anvers (Belgique), 1998.

[20] Charles Poynton. A guided tour to colour space. In *New Foundations for Video Technology*, San Francisco (USA), 1995.

[21] Yossi Rubner, Carlo Tomassi, and Leonidas Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2) :99–121, 2000.

[22] Shamik Sural, Gang Qian, and Sakti Pramanik. A histogram with perceptually smooth color transition for image retrieval. In *Fourth International Conference on Computer Vision, Pattern Recognition and Image Processing*, Durham, 2002.

[23] Michael Swain and Dana Ballard. Color indexing. *International Journal of Computer Vision*, 7(1) :11–32, 1991.

[24] Alain Tremeau, Christine Fernandez-Maloigne, et Pierre Bonton. *Image Numérique Couleur - De l'Acquisition au Traitement.* Dunod, Paris, 2004.

[25] Nicolas Vandenbroucke. *Segmentation d'Images Couleur par Classification de Pixels dans des Espaces d'Attributs Colorimétriques Adaptés. Application à l'Analyse d'Images de Football.* Thèse de doctorat, Université des Sciences et Technologies de Lille, Lille (France), 2000.

[26] Christopher C. Yang. Effects of coordinate systems on color image processing. Master's thesis, University of Arizona, Tucson, Arizona (USA), 1992.

[27] Xavier Zimmermann. Traitement d'images couleur pour le suivi de cibles : Application au positionnement d'un bras robotique. Rapport de DEA, Ecole Supérieure des Sciences Appliquées pour Ingénieur de Mulhouse, Mulhouse (France), 2004.

Annexes

Annexe A

Jeux de tests - Espaces couleur

A.1 Séquence n°1 - Déplacement

(a) Image n°1

(b) Image n°2

(c) Image n°3

(d) Image n°4

(e) Image n°5

(f) Cible

A.2 Séquence n°2 - Illumination

(a) Image n°1

(b) Image n°2

(c) Image n°3

(d) Image n°4

(e) Cible

A.3 Séquence n°3 - Illumination

(a) Image n°1

(b) Image n°2

(c) Image n°3

(d) Cible

A.4 Séquence n°4 - Déplacement

(a) Image n°1

(b) Image n°2

(c) Image n°3

(d) Image n°4

(e) Image n°5

(f) Cible

A.5 Résultats des tests avec le CHS

Espace	Quantification	Seq. 1	Seq. 2	Seq. 3	Seq. 4
RGB	3-3-2	100	100	0	100
	3-2-3	33	100	66	100
	2-3-3	33	100	0	100
CMY	3-3-2	100	100	0	100
	3-2-3	33	100	58	100
	2-3-3	33	100	0	100
CMY	3-3-2	58	100	0	100
	3-2-3	50	94	0	100
	2-3-3	58	94	0	100
XYZ	4-0-4	0	50	66	100
	3-2-3	33	75	25	100
YC_bC_r	2-3-3	42	100	25	100
	0-4-4	92	100	75	100
YIQ	2-3-3	100	100	33	95
	0-4-4	92	100	0	100
AC_1C_2	2-3-3	0	25	0	0
	0-4-4	42	75	0	80
AC_1C_2	2-3-3	33	75	58	40
	0-4-4	33	100	58	100
$L^*a^*b^*$	2-3-3	58	50	42	100
	0-4-4	0	25	8	100
HSV	4-2-2	100	100	92	100
	5-2-1	33	100	83	100
	3-3-2	58	100	66	100
$I_1I_2I_3$	2-3-3	58	100	58	95
	0-4-4	33	100	66	95

Annexe B

Jeux de tests - Estimation de taille

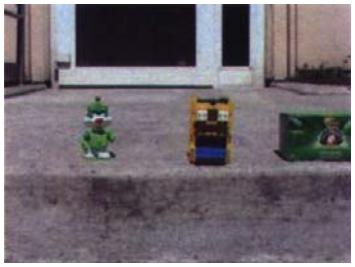
B.1 Séquence utilisée par le TROP

(a) Image n°1

(b) Image n°2

(c) Image n°3

(d) Image n°4



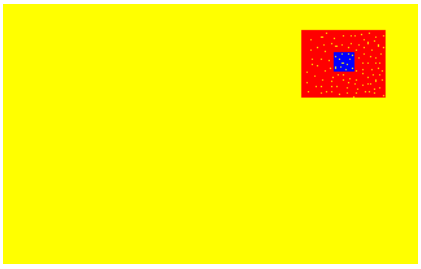
(e) Image n°5

(f) Cibles *RCX* et *FlunchBot*

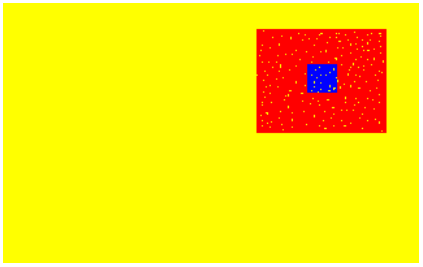
B.2 Analyse d'intervalle - Images synthétiques

B.2.1 Séquence

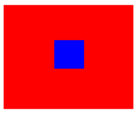
(a) Image n°1 — R=0.40



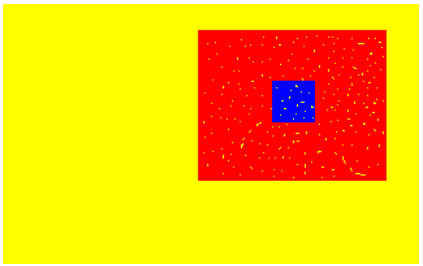
(b) Image n°2 — R=0.65



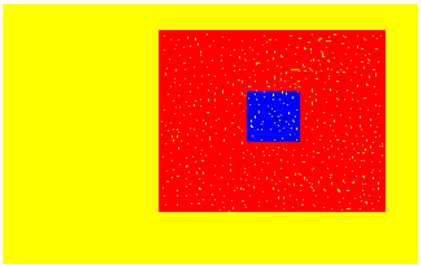
(c) Image n°3 — R=1.00



(d) Cible

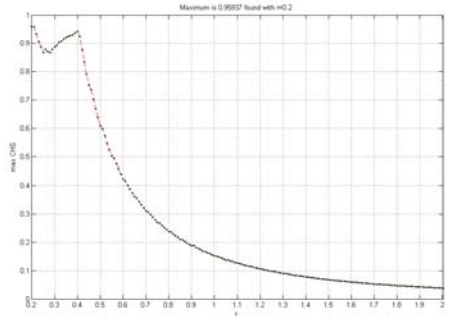


(e) Image n°4 — R=1.45

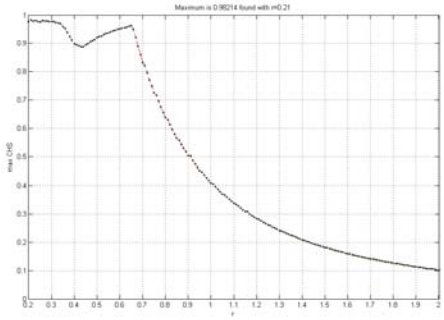


(f) Image n°5 — R=1.65

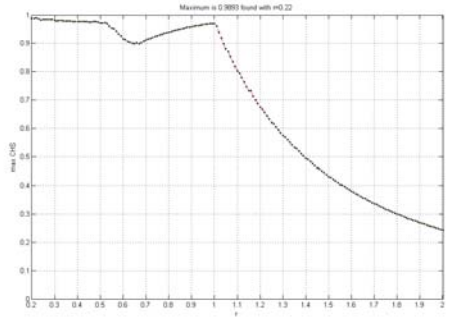
B.2.2 Résultats des tests



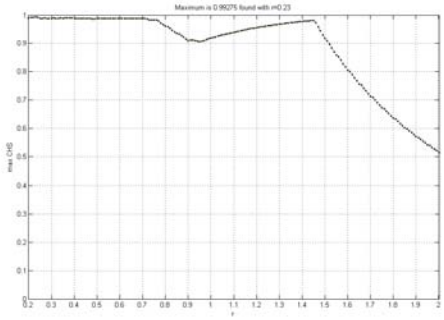
(a) Image n°1 — R=0.40



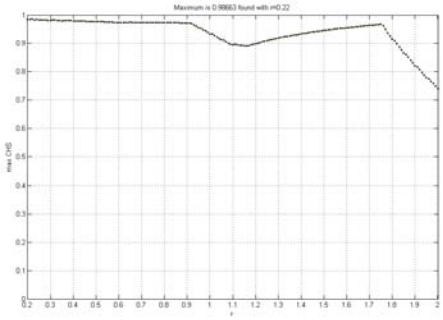
(b) Image n°2 — R=0.65



(c) Image n°3 — R=1.00



(d) Image n°4 — R=1.45



(e) Image n°5 — R=1.75

B.3 Analyse d'intervalle - Images réelles

B.3.1 Séquence

(a) Image n°1 — R=1.57

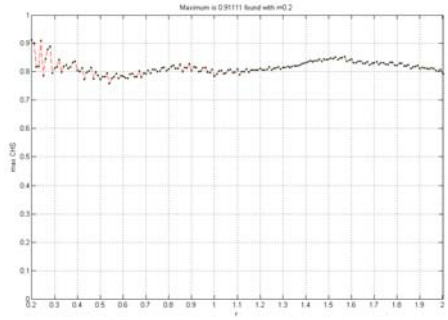
(b) Image n°2 — R=1.00

(c) Cible

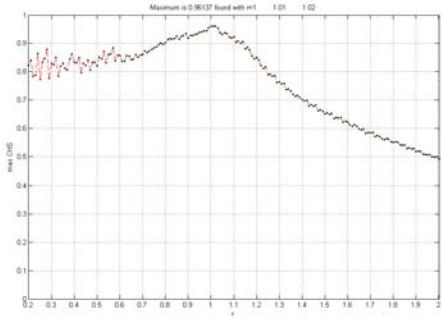
(d) Image n°3 — R=0.53

(e) Image n°4 — R=0.39

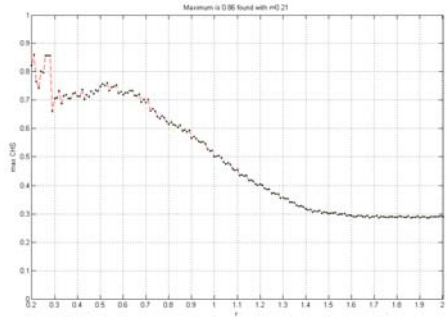
B.3.2 Résultats des tests



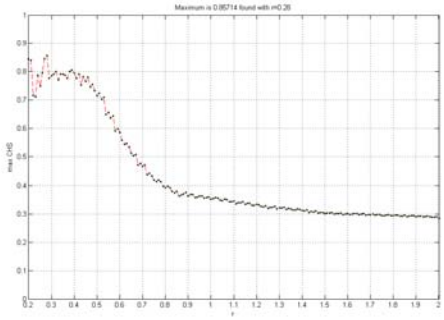
(a) Image n°1 — R=1.57



(b) Image n°2 — R=1.00



(c) Image n°3 — R=0.53



(d) Image n°4 — R=0.39

B.4 CHF - Séquence n°1

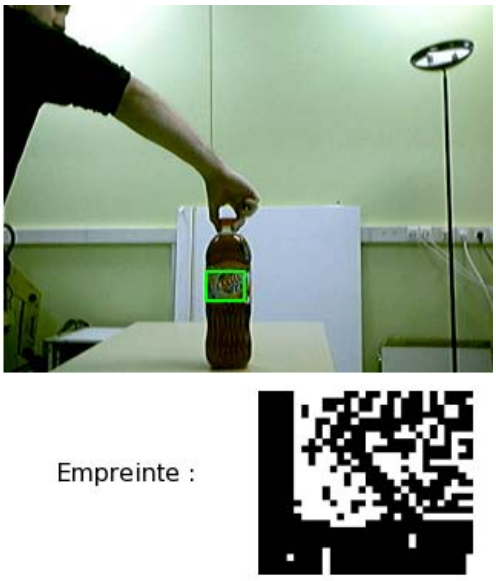
(a) Résultats du CHS sur la $1^{ère}$ image

(b) Résultats du CHF sur la $1^{ère}$ image

(c) Résultats du CHS sur la $2^{ème}$ image

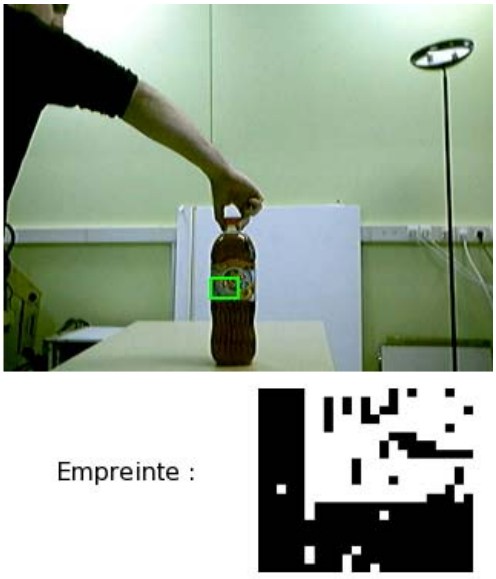
(d) Résultats du CHF sur la $2^{ème}$ image

B.5 CHF - Séquence n°2

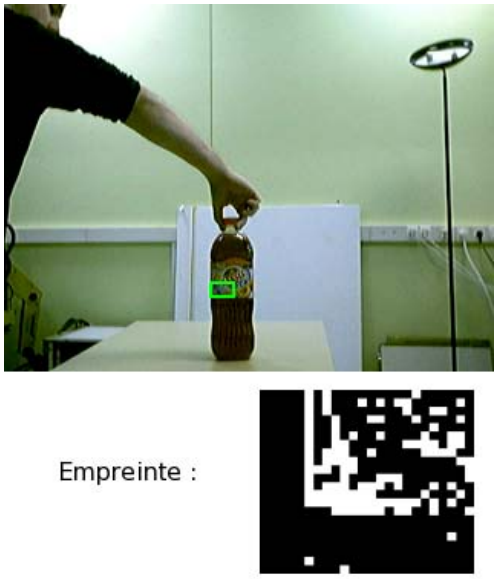


(a) $4^{ème}$ image de la vidéo

(b) $6^{ème}$ image de la vidéo



(c) $7^{ème}$ image de la vidéo



(d) $9^{ème}$ image de la vidéo

Annexe C

Jeux de tests - Indicateurs (A)

(e) Image n°1

(f) Image n°2

(g) Image n°3

(h) Image n°4

FIG. C.1 – Images du jeu de tests A

(a) Image n°5

(b) Image n°6

(c) Image n°7

(d) Image n°8

(e) Image n°9

(f) Image n°10

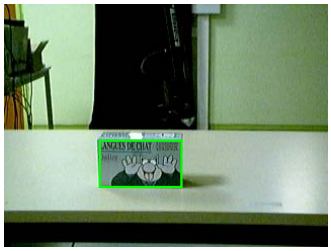
FIG. C.2 – Images du jeu de tests A

(a) Cible

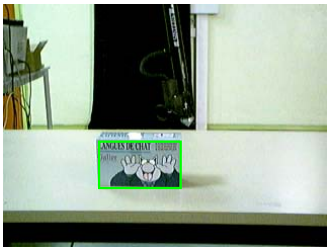
FIG. C.3 – Images du jeu de tests A

Annexe D

Jeux de tests - Indicateurs (B)

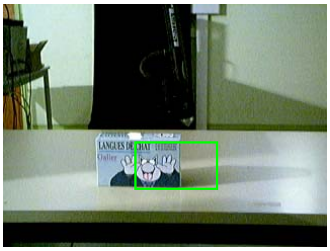


(a) Image n°1



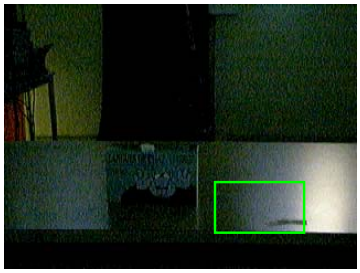
(b) Image n°2

(c) Image n°3



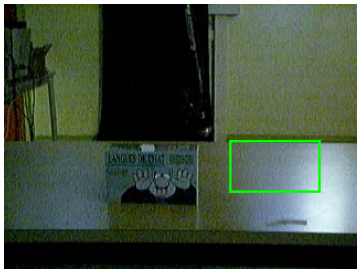
(d) Image n°4

FIG. D.1 – Images du jeu de tests B

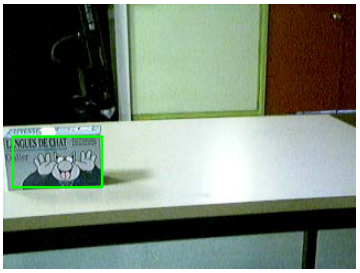


(a) Image n°5

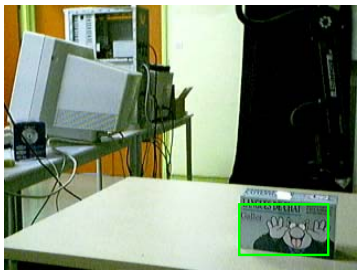
(b) Image n°6



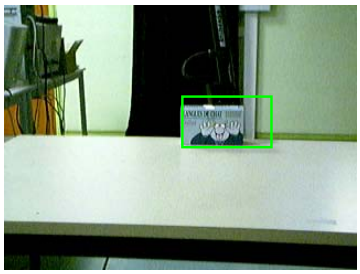
(c) Image n°7



(d) Image n°8

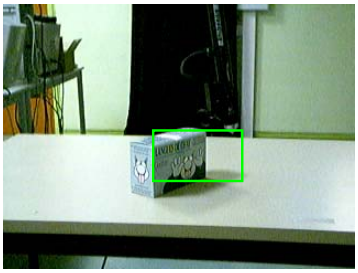


(e) Image n°9

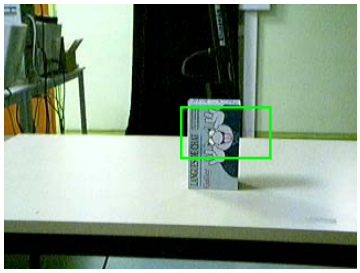


(f) Image n°10

FIG. D.2 – Images du jeu de tests B



(a) Image n°11



(b) Image n°12

(c) Cible

FIG. D.3 – Images du jeu de tests B

Annexe E

Application développée

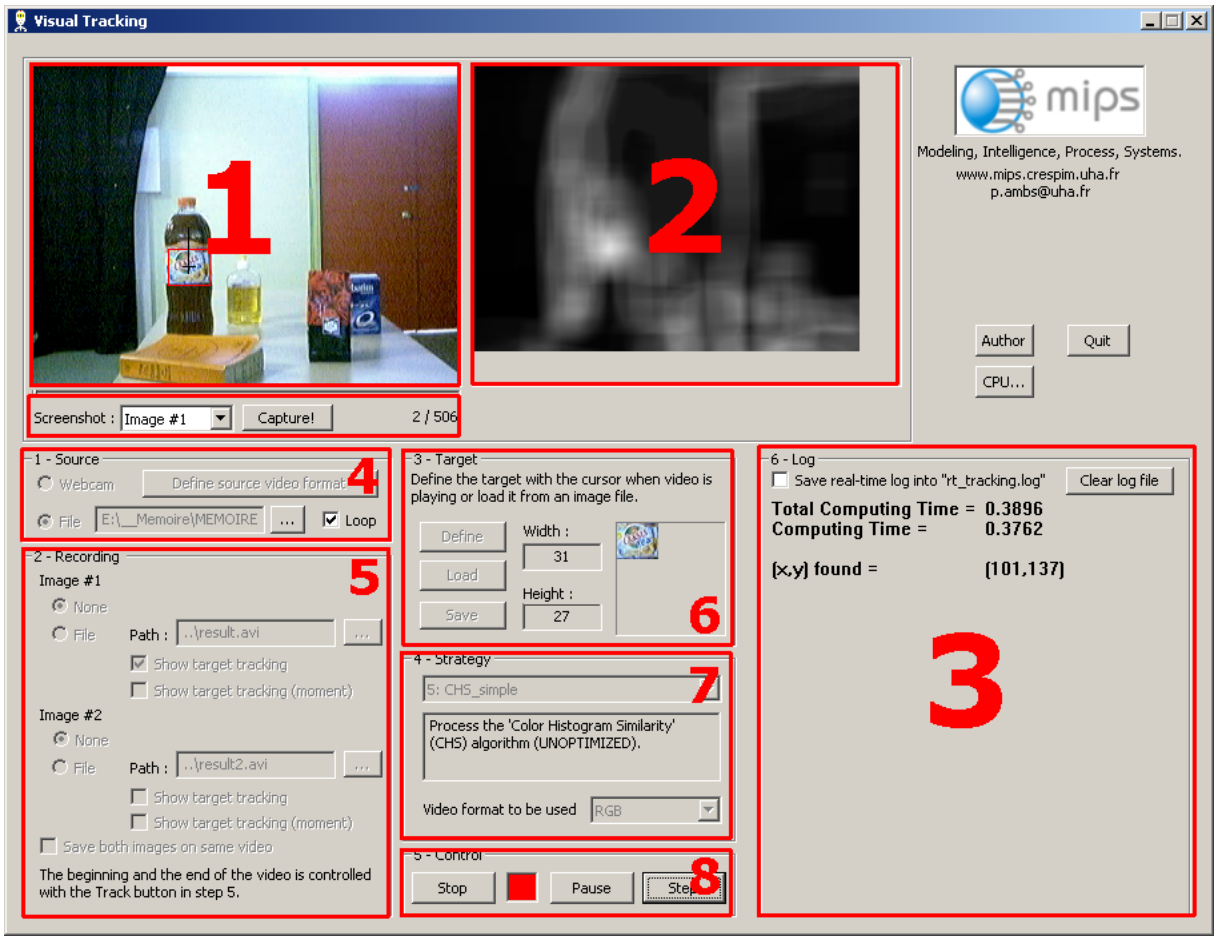


FIG. E.1 – Capture d'écran de l'application développée

Voici la description des zones du programme et des fonctionnalités qui leurs sont associées :

1. Zone de visualisation de la vidéo source utilisée (la vidéo peut être un fichier AVI, une suite de fichiers BMP ou les données d'une « webcam »). En-dessous, possibilité de capturer certaines images sous forme de fichier BMP.
2. Zone de visualisation des résultats (surface de similarité pour le CHS et empreinte pour le CHF).
3. Résultats supplémentaires tels que le temps de calcul, la position détectée, la taille détectée (pour le CHF), la validité des résultats, ... Possibilité de récupérer les données et de les exploiter sous Matlab grâce à un script.
4. Source de la séquence (fichier AVI, fichiers BMP ou une « webcam »). Possibilité de jouer une séquence en boucle.
5. Possibilité d'enregistrement de la séquence utilisée (quelle que soit la source) et des résultats produits.
6. Définition de la cible à localiser (définition sur la source ou chargement d'un fichier BMP). Possibilité d'enregistrer la cible dans un fichier BMP pour réutilisation ultérieure.
7. Définition de la stratégie (CHS, CHF, ...) et de l'espace couleur à utiliser (actuellement RGB ou HSV).
8. Contrôle de la séquence (commencer la localisation, arrêter la localisation, mettre en pause, ...). Possiblité d'avancer image par image (sauf lorsque la source est une « webcam »).